Data Aggregation and Demand Prediction

Maxime C. Cohen
McGill University, Montreal, maxime.cohen@mcgill.ca

Renyu Zhang
NYU Shanghai, 1555 Century Avenue, Shanghai, renyu.zhang@nyu.edu

Kevin Jiao
NYU Stern School of Business, New York, jjiao@stern.nyu.edu

We study how retailers can use data aggregation and clustering to improve demand prediction. High accuracy in demand prediction allows retailers to more effectively manage their inventory and mitigate stock-outs and excess supply. A typical retail setting involves predicting demand for hundreds of items simultaneously. Although some items have a large amount of historical data, others were recently introduced, and thus, transaction data could be scarce. A common approach is to cluster several items and estimate a joint model at the cluster level. In this vein, one can estimate some model parameters by aggregating the data from several items and other parameters at the individual-item level. We propose a practical method referred to as data aggregation with clustering (DAC), which balances the tradeoff between data aggregation and model flexibility. DAC allows us to predict demand while optimally identifying the features that should be estimated at the (i) item, (ii) cluster, and (iii) aggregate levels. We show that the DAC algorithm yields a consistent estimate, along with improved asymptotic properties relative to the decentralized method, which estimates a different model for each item. Using both simulated and real data, we illustrate DAC's improvement in prediction accuracy relative to common benchmarks. Interestingly, the DAC algorithm has theoretical and practical advantages and helps retailers uncover meaningful managerial insights.

Key words: Retail analytics, demand prediction, data aggregation, clustering

1. Introduction

Retailers routinely collect large volumes of historical data, which are used to improve future business practices, such as inventory management, pricing decisions, and customer segmentation. One of the most important data-driven tasks for retailers is to predict the demand for each stock-keeping unit (SKU). A common approach in practice is to classify SKUs into different departments (e.g., soft drinks) and sometimes even into sub-categories (e.g., a specific types of soft drinks) and then build predictive models accordingly. A typical demand prediction model is a regression specification with the sales (or logarithmic of the sales) as the outcome variable and price, seasonality, brand, color, and promotion as features. The model coefficients are then estimated using historical data.

In many retail settings, a subset of items has been offered for a long time (referred to as 'old items'), whereas other items were recently introduced. While the demand prediction for old items

is generally easy due to abundant data availability, accurately predicting the demand for newly introduced items with a limited number of historical observations is considerably more challenging. One might wonder how the available data of old items from the same department could be leveraged to enhance the prediction of new items. Indeed, SKUs in the same department often share similar characteristics and, hence, tend to be affected by a particular feature in a similar way. A prominent approach is to estimate certain coefficients at an aggregate level (i.e., by gathering the data across all SKUs and assuming a uniform coefficient). For example, it seems reasonable to believe that all items in the ice-cream category share the same seasonality pattern. Although this approach has been widely adopted in the retail industry, no rigorous empirical method has been developed to formalize how this data aggregation procedure should be applied for demand prediction. In this paper, we seek to bridge this gap by formalizing the tradeoff between data aggregation (i.e., pooling data from different items to reduce variance) and model flexibility (i.e., estimating a different model for each item to reduce bias) in a systematic fashion.

Due to insufficient data, the traditional approach of estimating a different model for each SKU is usually inefficient for new products or SKUs with noisy observations. This approach cannot identify the right aggregation level for each coefficient, and does not find the underlying cluster structure of the coefficients. Based on common clustering methods (e.g., k-means), we propose an efficient and integrated approach to infer the coefficient of each feature while identifying the right level of data aggregation based on the statistical properties of the estimated coefficients. Our method also allows us to incorporate multiple aggregation levels while preserving model interpretability. From a theoretical perspective, our method yields a consistent estimate, along with improved asymptotic properties. From a practical perspective, our method can easily be estimated using retail data and significantly improves out-of-sample prediction accuracy.

1.1. Main Results and Contributions

We study the tradeoff between data aggregation and model flexibility by optimally identifying the right level of aggregation for each feature, as well as the cluster structure of the items. We propose a practical method—referred to as the data aggregation with clustering (DAC) algorithm, which allows us to predict demand while optimally identifying the features that should be estimated at the (i) item, (ii) cluster, and (iii) aggregate levels. Our proposed algorithm first applies the maximum-likelihood estimation approach to estimate a different coefficient vector for each item (called the decentralized model). It then performs a hypothesis test (i.e., t-test) on the estimated coefficients from the decentralized model to identify the correct aggregation level for each feature. To characterize the cluster structure of the items, we apply the k-means method on the estimated coefficients from the decentralized model (as opposed to the features themselves).

We first characterize the DAC algorithm's theoretical properties. Specifically, we show that it yields a consistent estimate of the data aggregation levels, cluster structure, and feature coefficients. Thus, if the data has enough observations, one can correctly identify the underlying data generating process. In addition to this consistency result, we derive improved asymptotic properties—smaller variance and a tighter probabilistic bound—relative to the commonly-used ordinary least squares method applied in a decentralized fashion to each item. Furthermore, we show that if some items have abundant data while other items have limited data, our proposed algorithm improves the prediction accuracy for all items, with a more significant improvement for the items with limited data. Armed with these theoretical results, we then conduct extensive computational experiments based on both simulated and real data to illustrate the DAC algorithm's significantly improved prediction accuracy relative to 15 different benchmarks. Our results highlight the essential value of the DAC algorithm in better balancing the bias-variance tradeoff, resulting in more accurate demand prediction. Finally, we apply the DAC algorithm using two years of retail data and find that it can also help retailers uncover useful insights into the relationships between different items.

1.2. Related Literature

This paper is related to several literature streams, including prediction and clustering algorithms, retail operations, and demand forecasting.

Prediction and clustering algorithms: The problems of demand prediction and clustering have been extensively studied in the machine-learning (ML) literature. Bertsimas and Kallus (2020) combine ideas from ML and operations research to propose a new prediction method. The authors solve a conditional stochastic optimization problem by incorporating various ML methods, such as local regression and random forests. Donti et al. (2017) focus on developing new ML methods by training a prediction model to solve a nominal optimization problem. Although several studies have focused on general settings, it is difficult to apply existing methods to a retail setting where multiple levels of hierarchy may exist. Elmachtoub and Grigas (2017) propose a new idea called 'smart predict, then optimize' (SPO). SPO's key feature is that the loss function is computed based on comparing objective values generated using predicted and observed data. The authors then address the computational challenge and develop a tractable version of SPO. Jagabathula et al. (2018) propose a model-based embedding technique to segment a large population of customers into non-overlapping clusters with similar preferences. Liu et al. (2020) apply clustering techniques to predict the travel time of last-mile delivery services and optimize the order assignment. Our work is also related to the traditional clustering literature. Since the introduction of k-means by MacQueen et al. (1967), clustering algorithms have been extensively studied. In the context of assortment personalization, Bernstein et al. (2019) propose a dynamic clustering method to estimate customer

preferences. In our paper, we leverage some theoretical properties of the k-means clustering method and embed it as one of the key steps in our demand prediction algorithm.

Retail operations and demand forecasting: Retailers are always seeking ways to improve operational decisions, such as inventory replenishment, supply chain management, and pricing. These decisions rely heavily on accurate demand prediction. As reported by Cohen and Lee (2020), demand uncertainty is a major issue in designing efficient global supply chains. There is an extensive body of literature focused on developing methods for demand prediction in retail settings. Sophisticated models have been developed in the past two decades to manage the increasing volume of data collected by retailers. Marketing papers, such as Cooper et al. (1999), estimate econometrics models to draw managerial insights into the impact of retail promotions. In a similar vein, Van Heerde et al. (2000) and Macé and Neslin (2004) study pre- and post-promotion dips using linear regression models with lagged variables. Kök and Fisher (2007) develop a procedure to estimate substitution behavior in retail demand. Recent developments in demand prediction include the following three papers: Huang et al. (2014), who embed competitive information (including price and promotions) into demand prediction; Fildes et al. (2019), who suggest that promotional information can be quite valuable in improving forecast accuracy; and Huang et al. (2019), who further account for the impact of marketing activities. In the operations management community, demand prediction models are often used as an input to an optimization problem (e.g., Cohen et al. 2017, 2020). Specifically, Cohen et al. (2017) estimate a log-log demand model using supermarket data. The authors then solve the promotion optimization problem by developing an approximation based on linear programming. It has been shown in the retail operations literature that responding to accurate demand forecasts can substantially increase profits (Caro and Gallien 2010). Kesavan et al. (2010) show that incorporating the cost of goods sold, inventory, and gross margin information can substantially improve sales forecasting for retailers. In recent years, the amount of data available has grown exponentially, thus offering new opportunities for research on demand prediction (Feng and Shanthikumar 2018). In this context, our paper proposes a new demand prediction method that can efficiently aggregate data from multiple items to improve prediction accuracy.

A recent stream of papers integrate a clustering step into demand prediction. For instance, Baardman et al. (2017) propose an iterative approach to cluster products and leverage existing data to predict the sales of new products. Hu et al. (2017) propose a two-step approach to first estimate the product lifecycle and then cluster and predict. In our paper, however, the definition of clusters is fundamentally different. Unlike previous studies, our clustering is based on the estimated coefficients rather than the features' values. Furthermore, our model is flexible enough to account for different levels of data aggregation, whereas in previous studies, all features were essentially estimated at the cluster level. Allowing such flexibility is key to improving demand forecasting.

Structure of the paper. The remainder of the paper is organized as follows. In Section 2, we introduce our model and discuss the relevant computational challenges. We then describe the DAC algorithm in Section 3. Our analytical results are presented in Section 4. In Sections 5 and 6, we conduct computational experiments using simulated and real data, respectively. Our conclusions are reported in Section 7. The proofs of our analytical results are relegated to the Appendix.

2. Model

We introduce our demand prediction model under the generalized linear model (GLM) framework. Specifically, we consider a retail department (e.g., soft drinks or electronics) comprising n items (or SKUs). Each item has m historical observations (e.g., weekly sales transactions). We will show in Section 3 that our model and method can be generalized to a setting where different items have a different number of observations. For item i and observation j ($1 \leq i \leq n$ and $1 \leq j \leq m$), we denote the (log-of-)sales as $Y_{i,j}$ and the feature vector (e.g., price, promotion, seasonality, functionality, and color) as $X_{i,j} := (X_{i,j}^1, X_{i,j}^2, \ldots, X_{i,j}^d)' \in \mathbb{R}^d$. The feature set is denoted by $D := \{1, 2, \ldots, d\}$.

An important characteristic of our model is that a feature $l \in D$ may affect the demand/sales of an item at different data aggregation levels: (i) SKU, (ii) cluster, and (iii) department. More precisely, a feature may have the same impact on all items, captured by a uniform coefficient for all items in the department. We refer to such features as *shared* (i.e., department-level features), the set of which is denoted by D_s. Here, we consider a setting where all the items belong to the same department to be consistent with the usual business retail practice, where demand prediction is often performed for each department separately. We highlight, however, that our approach can be directly applied to a more general setting without a department structure. Alternatively, a feature may have a different impact on different items, captured by a different coefficient for each item. We refer to such features as *non-shared* (i.e., SKU-level features), the set of which is denoted by D_n. Finally, we assume that the items follow a cluster structure so that some features have the same impact on items within the same cluster and a different impact on items from a different cluster. This phenomenon is captured by a uniform coefficient for all items in the same cluster (the coefficients are different for each cluster). We refer to such features as cluster-level features, the set of which is denoted by D_c. We assume that the cluster structure is the same for all cluster-level features. However, our algorithm remains valid if the number of clusters or the cluster structure are different for different features, as we discuss in Section 3. The entire feature set, D, can be written as the union of three disjoint sets of features that affect the demand at different aggregation levels: $D = D_s \cup D_n \cup D_c$. The aggregation structure D_s, D_n, and D_c, and the cluster structure are unknown a priori and should be estimated from data.

We assume that the ground truth follows the GLM specification. Specifically, the observations are generated from an exponential family distribution that includes normal, binomial, and Poisson distributions as special cases. Based on the three data aggregation levels of the features, we have:

$$\mathbb{E}[Y_{i,j}] = g^{-1}\Big(\sum_{l\in D_s} X^l_{i,j}\beta^s_l + \sum_{l\in D_n} X^l_{i,j}\beta^n_{i,l} + \sum_{l\in D_c} X^l_{i,j}\beta^c_{\mathcal{C}(i),l}\Big), \qquad i=1,\dots,n \text{ and } j=1,\dots,m. \quad (1)$$

Here, $\mathcal{C}(i)\in\{1,\dots,k\}$ is the cluster that contains item i, and $g(\cdot)$ represents the link function that establishes the relationship between the linear predictor and the mean of the outcome variable. Furthermore, we use $\mathcal{C}_u$ to denote the set of items in cluster u, where $u\in\{1,2,\dots,k\}$ and $\{\mathcal{C}_1,\mathcal{C}_2,\dots,\mathcal{C}_k\}$ are a partition of the items $\{1,2,\dots,n\}$. There are many commonly-used link functions, and in practice, the function depends on the context. For example, if $Y_{i,j}$ is the number of sold units of item i in observation j, $g(\cdot)$ could be the identity function, and thus, the model reduces to a linear regression. On the other hand, if $Y_{i,j}$ is a binary variable, $g(\cdot)$ can be a logit function. Likewise, there exist other examples of link functions, such as logarithmic and inverse squared. Following the standard assumptions in the statistics and ML literature, we assume that all the observations are *independent* across both time periods and items. Extensions of the model and algorithm via vector auto-regression (resp. generalized least squares) to cases where observations may be correlated across time periods (resp. items) are discussed at the end of Section 3.

Based on Equation (1), we can characterize the aggregation levels of the three types of features. For a department-level feature $l\in D_s$, its coefficient β^s_l is shared among all the items. Thus, all the items in the department will have the same coefficient for this feature. In comparison, for a SKU-level feature $l\in D_n$, its coefficient $\beta^n_{i,l}$ will differ across items (i.e., $\beta^n_{i,l}\neq\beta^n_{i',l}$ for $i\neq i'$). Finally, for a cluster-level feature $l\in D_c$, all items in the same cluster will have the same coefficient (i.e., for all $i'\in\mathcal{C}(i)$, the coefficient of $X^l_{i',j}$ is equal to $\beta^c_{\mathcal{C}(i),l}$). Thus, the total number of coefficients for all n items is $d_x = n|D_n| + k|D_c| + |D_s| \le nd$. We use $n_{i,l}$ to denote the number of items that share the same coefficient as item i for feature l (i.e., $n_{i,l}=1$ if $l\in D_n$, $n_{i,l}=n$ if $l\in D_s$, and $n_{i,l}=|\mathcal{C}(i)|$ if $l\in D_c$). Estimating the coefficient of some features at a certain level of aggregation is common in practice. For example, retailers often estimate seasonality coefficients at the department level to avoid overfitting and capture the fact that the items follow the same seasonal patterns. Furthermore, when estimating the effect of promotions on demand, such as cannibalization or halo effects, one may cluster several items together because promotions often have a similar impact on a group of items. Assumption 1 below simplifies the exposition by avoiding the situation where two clusters have the same coefficient value.

ASSUMPTION 1. *(a) If a feature l is at the SKU level (i.e., $l\in D_n$), then $\beta_{i,l}\neq\beta_{i',l}$ for any pair of items i and i', that is, a SKU-level feature has a different effect on each item.*

(b) If cluster-level features exist (i.e., $D_c \neq \emptyset$), then, for $l \in D_c$, $\beta_{i,l} = \beta_{i',l}$ if and only if $\mathcal{C}(i) = \mathcal{C}(i')$, that is, a cluster-level feature has the same effect on items in the same cluster and a different effect on different clusters. Furthermore, each cluster has at least two items.

Our main goal is to accurately predict the dependent variable Y (in our case, weekly demand) given the feature vector X (e.g., price, promotion, seasonality, or color), assuming that the data generating process follows Equation (1). Before presenting our proposed estimation method, we first discuss the key challenge of fitting the model by using two intuitive methods. First, one could use the *constrained maximum-likelihood estimator* (MLE). This approach revises the standard MLE by adding the constraint that for items in the same cluster, the coefficients of the features at the cluster level should be the same. Likewise, the coefficients of a feature at the department level are the same across different items. Since we do not know a priori the aggregation level of each feature, the constrained MLE approach will involve solving an optimization problem with non-convex constraints, which is computationally prohibitive when the number of features is large. We provide a more detailed discussion of this approach applied to our problem in Appendix A.1.

A second possible approach is via *iterative optimization*, which is essentially based on the expectation-maximization (EM) algorithm. This approach introduces a binary decision variable to determine the aggregation level of each feature. To simultaneously estimate the aggregation level and the coefficient of the features, one can iteratively estimate the aggregation level (binary variable) and the coefficients using MLE. The iterative procedure will stop once the binary variables remain the same for two consecutive iterations. A similar iterative optimization approach was proposed by Baardman et al. (2017) to address the demand prediction problem with only cluster-level features (i.e., no department-level and no SKU-level features). In their setting, this iterative procedure was proved to converge to the true coefficients and cluster structure (i.e., the estimate is consistent). In our setting, however, the validity of this approach relies on the parameters' initialization. With inappropriate initial parameters, the procedure may get trapped in a local optimum. In other words, the approach proposed by Baardman et al. (2017) may not even be consistent. See Appendix A.2 for more details on the iterative optimization approach. Finally, an important limitation of both constrained MLE and iterative optimization is that neither method can simultaneously identify the right cluster structure and aggregation levels.

3. Data Aggregation With Clustering

As mentioned, simultaneously estimating the feature coefficients, aggregation levels, and cluster structure is computationally challenging and prone to substantial prediction errors. In this section, we propose a novel approach that allows us to (a) identify the correct level of aggregation for each feature, (b) find the underlying cluster structure of the SKUs, and (c) generate a consistent

estimate of the feature coefficients. Our method is entirely data-driven and can efficiently achieve these three goals in an integrated fashion while yielding an accurate demand prediction.

We begin our analysis by focusing on a (simple) special case of the model in Equation (1), where all the features are at the SKU level. In this case, the data-generating process can be written as

$$\mathbb{E}[Y_{i,j}] = g^{-1}\Big(\sum_{l\in D} X^l_{i,j} b_{i,l}\Big), \qquad i = 1,\dots,n \text{ and } j = 1,\dots,m. \tag{2}$$

Comparing Equations (1) and (2), we have $b_{i,l} = \beta^s_l$ for $l \in D_s$, $b_{i,l} = \beta^n_{i,l}$ for $l \in D_n$, and $b_{i,l} = \beta^c_{\mathcal{C}(i),l}$ for $l \in D_c$. We refer to model (2) as the *decentralized model* since we fit a model for each item in a decentralized fashion. Estimating the decentralized model is usually carried out through iterative re-weighted least squares, which ultimately lead to maximum-likelihood estimation (MLE). We assume that for each item, the decentralized model is well defined with a unique MLE solution, which is the case for commonly used GLMs, such as linear regression and logistic regression. Estimating the decentralized model can be decomposed into estimating one model for each item separately. Using the data of item i, we apply the MLE to find the estimated coefficients of this item, $\hat{b}_i := (\hat{b}_{i,1}, \hat{b}_{i,2}, \dots, \hat{b}_{i,d})' \in \mathbb{R}^d$:

$$\hat{b}_i \in \arg\max_{b_i} \sum_{j=1}^{m} \log \mathcal{L}(b_i | Y_{i,j}, X^1_{i,j}, X^2_{i,j}, \dots, X^d_{i,j}),$$

where $\mathcal{L}(b_i | Y_{i,j}, X^1_{i,j}, X^2_{i,j}, \dots, X^d_{i,j})$ is the likelihood function associated with the data $\{Y_{i,j}, (X^1_{i,j}, X^2_{i,j}, \dots, X^d_{i,j})\}$ and the coefficient vector $b_i = (b_{i,1}, b_{i,2}, \dots, b_{i,d})' \in \mathbb{R}^d$. We refer to the estimator $\hat{b} := (\hat{b}_1, \hat{b}_2, \dots, \hat{b}_n)$ as the *decentralized estimator*. We first show the following consistency property of the decentralized estimator $\hat{b}$, which directly follows from standard results in the MLE literature (e.g., Fahrmeir et al. 1985). This consistency property will be used as a building block for our subsequent analyses.

LEMMA 1. *The decentralized estimator $\hat{b}$ is consistent, that is, if $m \uparrow +\infty$, we have:*

- $\hat{b}_{i,l} \xrightarrow{p} \beta^s_l$ *for* $l \in D_s$;
- $\hat{b}_{i,l} \xrightarrow{p} \beta^n_{i,l}$ *for* $l \in D_n$;
- $\hat{b}_{i,l} \xrightarrow{p} \beta^c_{\mathcal{C}(i),l}$ *for* $l \in D_c$;

where $\xrightarrow{p}$ refers to convergence in probability.

Lemma 1 shows that with sufficiently many observations, we can consistently estimate the feature coefficients using the decentralized model. It is unsurprising that the decentralized estimator, $\hat{b}$, is consistent given that the decentralized model has a high amount of flexibility (which implies a low bias). Thus, if we have sufficiently many observations for each item, the prediction accuracy of the decentralized model will be high. However, two issues remain unaddressed with the decentralized

estimation: (i) how can we find the right aggregation level for each feature, and (ii) how can we identify the items' cluster structure. Furthermore, the decentralized estimation may suffer from overfitting and, hence, admit a high variance. We note that the number of coefficients in the ground truth from Equation (1) is strictly smaller than the number of coefficients generated by the decentralized estimator (i.e., $d_x = n|D_n| + k|D_c| + |D_s| < nd$, where $d = |D_n| + |D_c| + |D_s|$ is the dimension of the feature space). As shown in Section 6, by exploiting the data aggregation, we will efficiently mitigate the overfitting issue and substantially increase prediction accuracy.

To estimate the aggregation level and the underlying cluster structure based on Equation (1), we next introduce another special case of the model in which the data aggregation level and the cluster structure are known. We refer to this case as the *aggregated model* and call its MLE the *aggregated estimator*, which we denote by $\hat{\beta}$. For the aggregated model, we denote $\hat{\beta}^s_l$ as the estimated coefficient for a department-level feature, $\hat{\beta}^n_{i,l}$ for a SKU-level feature, and $\hat{\beta}^c_{\mathcal{C}(i),l}$ for a cluster-level feature. We are now ready to introduce the *data aggregation with clustering* (DAC) algorithm (described in Algorithm 1), which consistently estimates the coefficient of each feature for each item, as well as correctly identify the correct aggregation levels and the underlying cluster structure of the items.

Algorithm 1 DAC

Input: Estimated coefficient $\hat{b}_{i,l}$ and standard error ($\hat{SE}_{i,l}$) for each item i and feature l.

For each feature $l \in D$**,**

1: Fix an item 1. For all other items $i \neq 1$, compute the p-value based on the null hypothesis $H^0_{1,i}$ that $b_{1,l} = b_{i,l}$ (i.e., the coefficients of feature l are the same for item 1 and item i).
2: If $H^0_{1,i}$ is not rejected for all items, then feature l should be estimated at the aggregated level.
3: If $H^0_{1,i}$ is rejected for some items and validated for others, then feature l should be estimated at the cluster level. We then run a one-dimensional k-means on $\{\hat{b}_{i,l} : 1 \leq i \leq n\}$ and obtain the resulting clusters $\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_k$.
4: If $H^0_{1,i}$ is rejected for all items, then feature l should be estimated at the SKU level.
5: Obtain the aggregation level for each feature: $\hat{D}_n$, $\hat{D}_s$, and $\hat{D}_c$.
6: Fit an aggregated model to obtain the coefficients $\hat{\beta}$.

Output: (a) Aggregation levels: $(\hat{D}_n, \hat{D}_s, \hat{D}_c)$, (b) Cluster structure: $(\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_k)$, (c) Feature coefficients: $\hat{\beta}$.

By leveraging the consistent estimates obtained from the decentralized model (see Lemma 1), we can perform hypothesis testing to identify the correct data aggregation levels and cluster structure. Options for hypothesis testing include the Wald test and the likelihood-ratio test (see, e.g., Greene 2003). The main idea is as follows: if one cannot reject the null hypothesis that the estimated coefficients $\hat{b}_{i,l}$ and $\hat{b}_{i',l}$ are the same, then it is very likely that either items i and i' belong to the same cluster or that feature l is an aggregated-level feature. Another interesting characteristic of our method is that it uses the estimated coefficients as inputs to identify the cluster structure

of the items (see Step 3) as opposed to features as in traditional clustering algorithms. If some clusters do not agree for different features, one can implement a majority vote to find the optimal cluster structure. Alternatively, one can pool the cluster-level feature coefficients together and fit a multi-dimensional k-means. Either approach could produce a consistent estimate of the cluster structure. We also note that under Assumption 1, the pairwise hypothesis testing step has a time complexity of $O(nd)$. If we relax Assumption 1, the DAC can easily be adapted to run $O(n^2d)$ hypothesis tests instead of $O(nd)$. Furthermore, identifying the cluster-level features is even more efficient in practical implementations. Indeed, if we infer that the coefficients of a feature coincide for some items and differ for others, then this feature must be at the cluster level. Finally, we remark that the last step of the DAC algorithm to fit an aggregated model can be regularized using a Lasso or Ridge penalty to avoid unnecessary features and mitigate overfitting. This would be especially useful if some of the features are correlated, which is common in retail settings.

We next show that with sufficiently many observations, the DAC algorithm can consistently identify the aggregation level of each feature, as well as the underlying cluster structure.

PROPOSITION 1. *The DAC algorithm outputs a consistent estimate of the aggregation level for each feature and the underlying cluster structure of the items:*

$$\lim_{m\uparrow\infty} \mathbb{P}\Big[(\hat{D}_n, \hat{D}_s, \hat{D}_c) \neq (D_n, D_s, D_c) \text{ or } (\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_k) \text{ is not a permutation of } (\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k)\Big] = 0.$$

As shown in Appendix B.2, the main idea behind the proof of Proposition 1 is to leverage the consistency of the decentralized estimator. The estimated coefficients in the decentralized model will eventually converge to their true values and, hence, allow us to accurately learn the aggregation level and the cluster structure. We highlight that the choice of item 1 in the first step of DAC is without loss of generality. Indeed, choosing another item in this step will also produce a consistent estimate of the aggregation levels $(\hat{D}_n, \hat{D}_s, \hat{D}_c)$, the cluster structure $(\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_k)$, and the feature coefficients $\hat{\beta}$. This follows from the fact that all the cluster-level features share the same cluster structure as the items (i.e., $(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k)$). Thus, a feature l is identified to be at the cluster level for item i if and only if it is also identified to be at the cluster level for another item $i' \neq i$.

Several remarks are in order regarding the DAC algorithm and its consistency. First, Proposition 1 shows the effectiveness of Algorithm 1 under the asymptotic regime (i.e., the sample size m goes to infinity). For the case where the data sample size m is small (e.g., a new item with limited data), the model's variance will be high. In this case, the DAC algorithm, based on hypothesis testing, may misspecify a SKU-level feature as a cluster-level or an aggregated-level feature. Such a misspecification may result in biases and, ultimately, in prediction errors. To address this misspecification issue, we adopt a cross-validation procedure to fine-tune several hyper-parameters, including the significance level for hypothesis testing and the threshold for classifying each feature

into clustering and department levels (see Sections 5 and 6 for the implementation details). More precisely, this procedure aims to control for Type-II errors of misclassifying SKU-level features as cluster- or department-level features. Using both simulated and real data, in Sections 5 and 6, we show that our proposed DAC algorithm outperforms a multitude of well-established benchmarks used in the literature and in practice. We also examine the case with new items that have limited data availability. Our theoretical results (Proposition 4) and our computational results (Figure 5) show that our DAC algorithm can efficiently pool and leverage the data, while substantially improving the prediction accuracy for *all* items regardless of their data availability. More importantly, the accuracy improvement is more substantial for items with limited data.

Second, we assumed that the number of clusters k is known upfront and that both the cluster number and the cluster structure with respect to the items are the same across different features. Since both assumptions may not be valid in practice, we next propose a way to relax them. Specifically, for the practical implementation of the DAC algorithm, k can be treated as a hyper-parameter and is, thus, determined via a standard cross-validation procedure (see Section 6). Furthermore, the algorithm can also be applied to a setting where the cluster structure is not the same for the different features, while still yielding consistent estimates. If the number of clusters is not necessarily identical for each feature (i.e., the number of clusters for feature l is k_l), the DAC algorithm can be adapted to incorporate the cross-validation on k_l for each $1 \leq l \leq d$. The proof of Proposition 1 can then be extended to show that such a modified version of Algorithm 1 preserves consistency.

Finally, we assume in our base model that all the observations are independent across time periods and items. Such an independence assumption can also be relaxed through the vector auto-regression (VAR) model for the case with correlations across time periods and through the generalized least squares (GLS) model for the case with correlations across items. To conclude this section, we discuss how the DAC algorithm can be generalized to these two cases.

Correlation across time periods. To account for correlations across time periods, we assume that the observations $\{1, 2, \ldots, m\}$ are ranked in chronological order. More precisely, the data collected in period j are indexed as observation j. We consider the following VAR model specification:

$$Y_{i,j} = \alpha_i Y_{i,j-1} + \sum_{l \in D_s} X^l_{i,j} \beta^s_l + \sum_{l \in D_n} X^l_{i,j} \beta^n_{i,l} + \sum_{l \in D_c} X^l_{i,j} \beta^c_{\mathcal{C}(i),l} + \epsilon_{i,j}, \ i = 1, \ldots, n \text{ and } j = 2, \ldots, m, \quad (3)$$

where $\alpha_i Y_{i,j-1}$ is the auto-regression term, β^s_l, $\beta^n_{i,l}$, and $\beta^c_{\mathcal{C}(i),l}$ are defined in the same fashion as in the base model, and $\epsilon_{i,j}$ are the *i.i.d.* unobservable error terms uncorrelated with the features X (i.e., $\mathbb{E}[\epsilon_{i,j}|X] = 0$). Applying the standard VAR estimation approach (see, e.g., Chapter 19 of Greene 2003), we could follow the same procedure as Algorithm 1 to estimate the decentralized

model, conduct hypothesis tests, cluster the items, and, finally, estimate the aggregated model. All the results presented in this paper remain valid under this VAR setting.

Correlation across items. To account for correlations across items, we consider the following GLS model specification:

$$Y_{i,j} = \sum_{l \in D_s} X_{i,j}^l \beta_l^s + \sum_{l \in D_n} X_{i,j}^l \beta_{i,l}^n + \sum_{l \in D_c} X_{i,j}^l \beta_{\mathcal{C}(i),l}^c + \epsilon_{i,j},\ i = 1, \ldots, n \text{ and } j = 1, \ldots, m, \tag{4}$$

where β_l^s, $\beta_{i,l}^n$, and $\beta_{\mathcal{C}(i),l}^c$ are defined in the same fashion as in the base model, and $\epsilon_{i,j}$ are the unobservable error terms with $\mathbb{E}[\epsilon_{i,j}|X] = 0$, but $\mathbb{E}[\epsilon_{i,j}\epsilon_{i',j}|X] \neq 0$ for $i \neq i'$. For this model, we could slightly modify the DAC algorithm to estimate the coefficients. More specifically, we follow the same procedure as in Steps 1–5 of Algorithm 1 to estimate the decentralized model, conduct hypothesis tests, and cluster the items. However, the final step (Step 6) that estimates the aggregated model should be adjusted to apply the GLS (instead of OLS) approach (see, e.g., Chapter 10 of Greene 2003). All the results presented in this paper remain valid under this GLS setting. If there are correlations across both time periods and items, we could combine the VAR and GLS frameworks and modify the DAC algorithm accordingly to yield consistent estimates of the model coefficients, data aggregation levels, and cluster structure.

4. DAC's Theoretical Properties

The remainder of this paper is devoted to characterizing the benefits of performing the pairwise tests and the clustering algorithm relative to the decentralized model. Specifically, we present three sets of analyses from different perspectives: (a) analytical comparisons between the aggregated and decentralized models, which highlight the value of data aggregation through efficient clustering; (b) simulation studies of the DAC algorithm versus several benchmarks, which show that the DAC algorithm can successfully identify and leverage data aggregation and the cluster structure; and (c) implementation of the DAC algorithm using real retail data, which showcases the practical value of the DAC algorithm in improving demand prediction accuracy.

In this section, we examine the value of data aggregation through efficient clustering from a theoretical perspective by showing several benefits of the aggregated model relative to the decentralized model. To convey the DAC's benefits, we first observe that if the true data-generating process has different aggregation levels across different features, the decentralized model assumes an overly complex model and, hence, will be prone to overfitting. To formalize this intuition, we leverage the asymptotic normality property of the MLE. Before presenting the result, we first introduce some notation. We define the log-likelihood of item i with the data sample $\{(Y_{i,j}, X_{i,j}^1, \ldots, X_{i,j}^d) : j = 1, 2, \ldots, m\}$ as $l_i(\beta_i; m) := \frac{1}{m}\sum_{j=1}^m \log \mathcal{L}_i(\beta_i | Y_{i,j}, X_{i,j}^1, \ldots, X_{i,j}^d)$. Thus, the decentralized estimator is $\hat{b}_i(m) = \arg\max_{\beta_i} l_i(\beta_i; m)$, where the parameter m captures its dependence on the sample size.

We also denote the gradient and Hessian of the log-likelihood function associated with item i by $\nabla l_i(\beta_i;m)$ and $\nabla_2 l_i(\beta_i;m)$, respectively. The Fisher information matrix with respect to the decentralized model of item i is thus given by $\mathcal{I}_i(\beta_i) = -\mathbb{E}[\nabla_2 l_i(\beta_i;1)]$, where the expectation is taken with respect to the true value of β_i and the true distribution of $(Y_{i,1}, X^1_{i,1}, \ldots, X^d_{i,1})$. We then have $\lim_{m\to+\infty} \nabla_2 l_i(\beta_i;m) = -\mathcal{I}_i(\beta_i)$. We next define the log-likelihood of all the items as follows:

$$l(\beta;m) = \sum_{i=1}^{n} l_i(\beta_i;m) := \sum_{i=1}^{n} \frac{1}{m} \sum_{j=1}^{m} \log \mathcal{L}_i(\beta_i | Y_{i,j}, X^1_{i,j}, \ldots, X^d_{i,j}).$$

The aggregated estimator is then defined by $\hat{\beta}(m) = \arg\max_{\beta\in\Xi} l(\beta;m)$, where the feasible parameter set Ξ is defined as $\Xi = \{\beta : \beta^l_i = \beta^l_{i'}$ if (a) $l \in D_s$ or (b) $l \in D_c$ and $i' \in \mathcal{C}(i)\}$. We denote the gradient and Hessian of $l(\beta;m)$ as $\nabla l(\beta;m) = \sum_{i=1}^n \nabla l_i(\beta;m)$ and $\nabla_2 l(\beta;m) = \sum_{i=1}^n \nabla_2 l_i(\beta;m)$, respectively. The Fisher information matrix with respect to the aggregated model is thus given by $\mathcal{I}(\beta) = -\mathbb{E}[\nabla_2 l(\beta;1)] = -\sum_{i=1}^n \mathbb{E}[\nabla_2 l_i(\beta_i;1)] = \sum_{i=1}^n \mathcal{I}_i(\beta_i)$, where the expectation is taken with respect to the true value of β and the true distribution of $(Y_{i,1}, X^1_{i,1}, \ldots, X^d_{i,1})$. Furthermore, we have $\lim_{m\to+\infty} \nabla_2 l(\beta, m) = -\mathcal{I}(\beta)$. We are now ready to compare the (asymptotic) variance of the aggregated and decentralized models. We use $\mathsf{Var}(\cdot)$ to denote the variance operator.

PROPOSITION 2. *For the aggregated and decentralized models, the following statements hold:*
(a) $\hat{\beta}(m)$ and $\hat{b}(m)$ converge to the following asymptotic distributions as $m \to \infty$,

$$\sqrt{m}(\hat{\beta}(m) - \beta) \xrightarrow{d} N(0, \mathcal{I}(\beta)^{-1}),$$
$$\sqrt{m}(\hat{b}_i(m) - \beta_i) \xrightarrow{d} N(0, \mathcal{I}_i(\beta_i)^{-1}), \text{ for } i = 1, 2, \ldots, n,$$

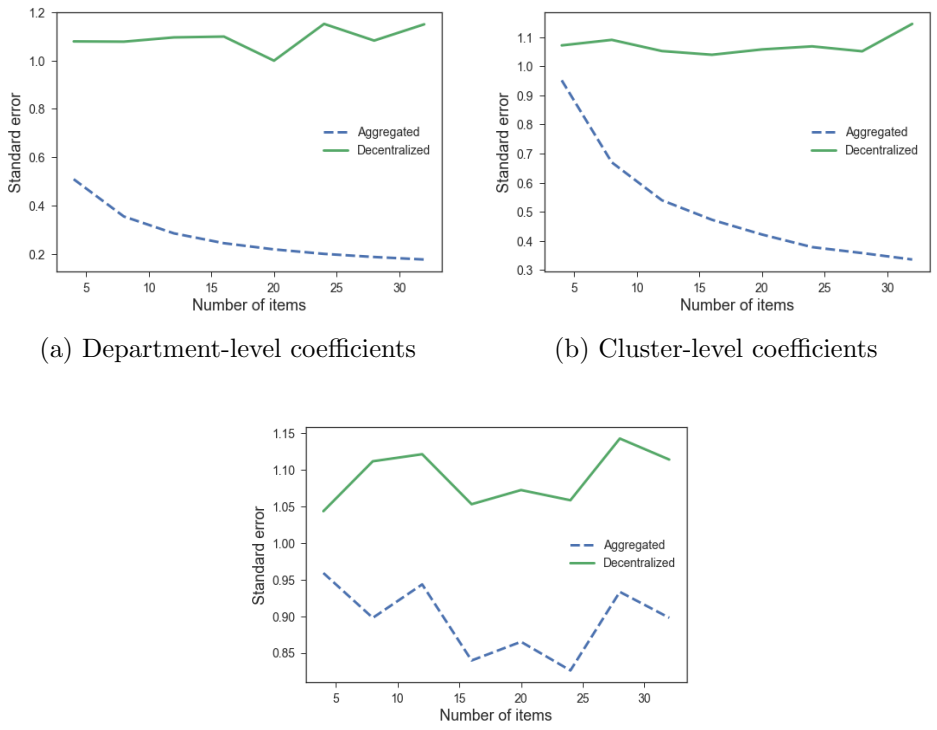
where $\xrightarrow{d}$ refers to convergence in distribution.

(b) There exists a constant $\kappa_{i,l} > 0$ for any (i,l), such that

$$\lim_{m\to+\infty} m \cdot n_{i,l} \cdot \mathsf{Var}(\hat{\beta}_{i,l}(m)) = \frac{1}{n_{i,l}} \sum_{i'\in\mathcal{C}(i,l)} \kappa_{i',l}, \text{ for } i = 1, 2, \ldots, n, \ l = 1, 2, \ldots, d,$$
$$\lim_{m\to+\infty} m \cdot \mathsf{Var}(\hat{b}_{i,l}(m)) = \kappa_{i,l}, \text{ for } i = 1, 2, \ldots, n, \ l = 1, 2, \ldots, d,$$

where $\mathcal{C}(i,l)$ is the set of items whose feature l has the same coefficient as item i.

As expected, when the number of observations m becomes large, the variance of the estimated coefficients of both models will shrink to zero. What makes the aggregated model more powerful is its ability to pool the data from different items, and thus, it further reduces the variance of the estimates, as shown in Proposition 2(b). Note that $\frac{1}{n_{i,l}} \sum_{i'\in\mathcal{C}(i,l)} \kappa_{i',l}$ is the average variance of the coefficient of feature l for items in $\mathcal{C}(i,l)$. Since $n_{i,l} \geq 2$ for $l \in D_s \cup D_c$, the aggregated estimation yields a smaller asymptotic variance relative to the decentralized estimation for the coefficients of



(a) Department-level coefficients (b) Cluster-level coefficients

(c) SKU-level coefficients

Figure 1 Comparison of standard errors for the aggregated and decentralized models.

features at the department or cluster levels. In this case, the aggregated model will use at least twice as many observations as the decentralized model to estimate the coefficient of this feature. Hence, the variance will shrink faster, especially when $n_{i,l}$ is large. In practice, a typical retail department consists of a large number of items ($n > 100$), so the aggregated model can be much more efficient than the decentralized model for the cluster- and department-level features. Finally, we remark that Proposition 2 holds for both the case with a diagonal information matrix (i.e., the off-diagonal entries are all 0) and the case with a general information matrix, so the estimated coefficients could be correlated in the asymptotic regime.

For the non-asymptotic regime where the sample size m is small, we next convey the efficiency improvement of the aggregated estimator relative to the decentralized estimator by numerically computing the standard error of the estimated coefficients at different aggregation levels. In Figure 1, we consider a simple illustrative example where we fix the number of observations for each item ($m = 50$) and the number of clusters ($k = 4$).[1] For each value of n, we generate 100 independent instances to compute the average standard error of the estimated coefficients for both the aggregated and decentralized estimators (for each type of feature). As n increases, the standard

[1] Parameters for Figure 1: $d = 3$ (one feature at each level), β is obtained from a uniform $[-2, 2]$, X from a uniform $[0, 1]$, and $\sigma^2 = 0.1$ (for more details, see Section 5). For each n, we generate the data and estimate both models.

errors of the estimated coefficients for the department and cluster levels decrease monotonically for the aggregated model. In contrast, n does not affect the standard errors for the decentralized model. The estimated coefficients improve substantially as n increases. These plots demonstrate the significant efficiency improvement (i.e., reducing the variance) of the aggregated model relative to the decentralized model when there are department- and cluster-level features. We next derive probabilistic concentration bounds for the mean squared error under the OLS setting for both the decentralized and aggregated models.

PROPOSITION 3. *Under the OLS setting, we have*

$$\mathbb{P}\Big(\frac{||X\beta - X\hat{\beta}||_2^2}{n \times m} \leq \frac{\sigma^2\big[2\sqrt{\gamma d_x} + 2\gamma + d_x\big]}{n \times m}\Big) \geq 1 - \exp(-\gamma)$$

for the aggregated model, and

$$\mathbb{P}\Big(\frac{\sum_{i=1}^n ||X_i b_i - X_i \hat{b}_i||_2^2}{n \times m} \leq \frac{\sigma^2\big[2\sqrt{\gamma\Big(nd\Big)} + 2\gamma + nd\big]}{n \times m}\Big) \geq 1 - \exp(-\gamma)$$

for the decentralized model. Furthermore, when $d_x \leq nd - 2\Big(\sqrt{\gamma nd} + \sqrt{\gamma nd - 2\gamma\sqrt{nd} - \gamma^2}\Big)$*, there exists a threshold value* Γ *such that*

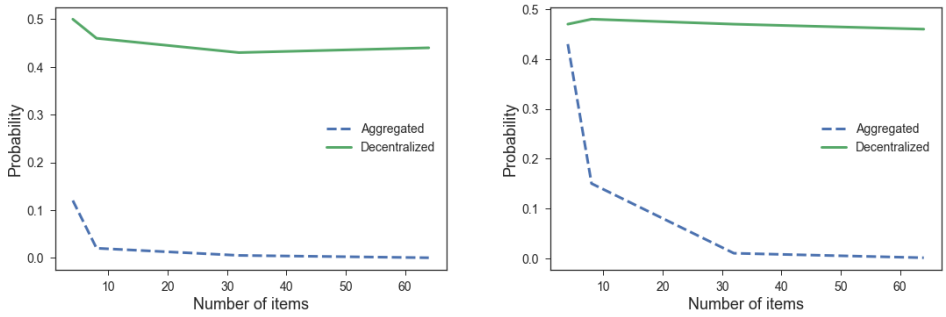
$$\mathbb{P}\Big(\frac{||X\beta - X\hat{\beta}||_2^2}{n \times m} \leq \Gamma \text{ and } \Gamma \leq \frac{\sum_{i=1}^n ||X_i b_i - X_i \hat{b}_i||_2^2}{n \times m}\Big) \geq 1 - \exp(-\gamma), \tag{5}$$

which implies that the aggregated model outperforms the decentralized model with high probability (if we set γ *large enough).*

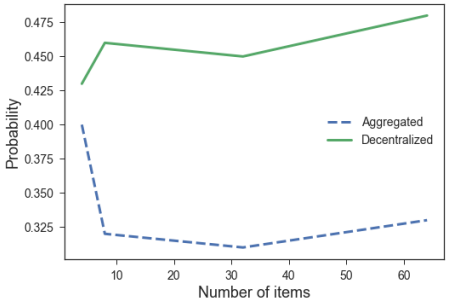
The parameter γ is a positive constant that captures the desired probability bound. Recall that $d_x = n|D_n| + k|D_c| + |D_s|$ corresponds to the total number of coefficients in our model. Proposition 3 implies that with high probability, the estimation error of the aggregated model is smaller relative to the decentralized model, as long as the number of coefficients is small. Note that condition (5) is well defined only when $\gamma \leq nd - 2\sqrt{nd}$. For a GLM, such as logistic regression, we can use asymptotic normality and localization techniques to develop similar concentration bounds in the asymptotic regime (i.e., when the sample size m is large). Table 1 illustrates the result of Proposition 3 (i.e., how large d_x can be relative to the total number of features nd).

Table 1 Maximum value of d_x.

γ	Probability	nd	d_x^*
1	0.632	500	412
2	0.865	1,000	824
5	0.993	3,000	2,514
10	0.999	5,000	4,112



(a) Department-level coefficients (b) Cluster-level coefficients



(c) SKU-level coefficients

Figure 2 Comparison of large estimation error probability for aggregated and decentralized models.

As we can see from Table 1, the upper bound on d_x (denoted by d_x^*) is relatively easy to satisfy. For example, when $nd = 1,000$, it means that for the decentralized model, one needs to estimate 1,000 parameters. In this case, the aggregated model will outperform the decentralized model with probability (at least) 0.865, as long as the number of features is less than 824. Suppose now that the number of parameters in the original model is large (e.g., $nd = 5,000$). As long as there is a non-negligible number of department- and cluster-level features to reduce the number of coefficients to 4,112, then the aggregated model will outperform the decentralized model with a probability very close to 1. This ultimately illustrates the power of aggregating data and reducing the dimensionality in order to improve prediction accuracy.

For a non-linear GLM model (e.g., logistic regression), since the prediction accuracy does not have a closed-form expression, we study the probabilistic bound computationally. For a given coefficient, we compute $\mathbb{P}(|\hat{\beta}_{i,l} - \beta_{i,l}| > \eta)$, which measures the confidence level of the estimate. As we can see from Figure 2, regardless of the aggregation level, the probability that the estimated coefficient is far from the true value is lower for the aggregated model relative to the decentralized model, especially when n is large.[2]

[2] Parameters for Figure 2: $m = 200$, $k = 4$, $d = 3$ (one feature at each level), β is obtained from a uniform $[-2, 2]$, X from a uniform $[0, 1]$, and $\eta = 0.4$. For each n, we generate the data and fit both models. Since we generate 100 independent instances for each n, we count the number of instances where $|\hat{\beta} - \beta| > \eta$ to compute the probability.

We next study an important generalization of our model with two items, each with a different number of observations. This setting fits the scenario where one item has been offered for a long time (and, hence, has abundant data), and the other item is new to the market (and, hence, has limited data). More specifically, we assume that item 1 has m observations, whereas item 2 has τm observations, where $\tau \geq 1$ captures the relative sample size ratio of the two items. Since we only have two items, a feature is either at the individual level or at the department level. We define $d_n := |D_n|$ as the number of features at the individual level and $d_s := |D_s|$ as the number of features at the aggregate level. To highlight the main intuition without getting trapped in the technical details, we focus on the OLS setting with independent features. Namely, we assume that for items $i = 1, 2$ and for all the data points j, $X^l_{i,j}$'s are *i.i.d.* with mean 0 and variance 1. The underlying data-generating process is specified as follows:

$$Y_{i,j} = \sum_{l \in D_n} \beta^n_{i,l} X^l_{i,j} + \sum_{l \in D_s} \beta^s_l X^l_{i,j} + \epsilon_{i,j},$$

where the error term $\epsilon_{i,j}$ is *i.i.d.* with mean zero and variance $\mathbb{E}[\epsilon^2_{i,j}] = \sigma^2_\epsilon$ for items $i = 1, 2$ and for all observations j. In the matrix form, we have, for item i, $Y_i = X_i \beta_i + \epsilon_i$, where $\beta_i = (\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,d})'$ is the coefficient vector, and $\epsilon_1 = (\epsilon_{1,1}, \epsilon_{1,2}, \ldots, \epsilon_{1,m})'$ (resp. $\epsilon_2 = (\epsilon_{2,1}, \epsilon_{2,2}, \ldots, \epsilon_{2,\tau m})'$) is the error term vector for item 1 (resp. item 2). The following proposition demonstrates the effectiveness of the DAC algorithm (in the asymptotic regime) for the two-item case, one with abundant data and the other one with limited data.

Proposition 4. *Consider an OLS setting with two items. Assume that item 1 has m i.i.d. observations, and item 2 has τm (where $\tau \geq 1$) i.i.d. observations, with the same feature distributions for both items. Define the generalization error for items $i = 1, 2$ under the estimation algorithm π as $\mathsf{e}_i(\pi) := \mathbb{E}[Y_i - \sum_{l \in D} \hat{\beta}_{i,l}(\pi) X_l]^2$, where $\hat{\beta}_{i,l}(\pi)$ is the estimated coefficient of feature l for item i, and the expectation is taken with respect to the training data and (Y_i, X_i) following the same distribution as the training set. We then have*

$$\lim_{m \to +\infty} m \cdot (\mathsf{e}_1(Dec) - \mathsf{e}_1(Agg)) = \left(1 - \frac{1}{1+\tau}\right) \cdot d_s \cdot \sigma^2_\epsilon,$$
$$\lim_{m \to +\infty} m \cdot (\mathsf{e}_2(Dec) - \mathsf{e}_2(Agg)) = \left(\frac{1}{\tau} - \frac{1}{1+\tau}\right) \cdot d_s \cdot \sigma^2_\epsilon,$$

where Dec and Agg refer to the decentralized and aggregated approaches, respectively.

Proposition 4 shows that when the sample size becomes large, the generalization error of either the item with abundant data or the item with less data will be lower under the aggregated approach relative to the decentralized approach. Furthermore, such data aggregation reduces the generalization error more significantly for the item with less data (item 1). As expected, the prediction

accuracy improvement of the DAC algorithm is strengthened when the number of features at the aggregate level, d_s, is larger. The proof of Proposition 4 is built upon the bias-variance decomposition of a linear regression model. We highlight that the DAC algorithm improves the prediction accuracy for all the items regardless of their data availability. This insight continues to hold for the case where the number of items is larger than two and when some features may be at the cluster level. We show the robustness of this result via extensive computational experiments in Section 5.2. The result of Proposition 4 also indicates that the error reduction for item 1 (fewer data) increases concavely with τ, whereas the error reduction for item 2 (abundant data) decreases convexly with τ. Indeed, when τ increases, item 1 can successfully leverage the abundant data from item 2 with diminishing marginal returns, whereas item 2 could extract less additional information from item 1's data.

To conclude this section, we remark that all our analysis has focused on comparing the aggregated and decentralized models. We note that in practice, the data aggregation level and the cluster structure are not known a priori, but are identified via hypothesis testing and by the k-means step of the DAC algorithm. As a result, one could expect additional biases and increased variances for the algorithm relative to the aggregated model. Ultimately, one may question whether the value of data aggregation and clustering remains significant for the DAC algorithm. Our simulation and real data studies (Sections 5 and 6) clearly convey that our proposed DAC algorithm efficiently identifies and leverages data aggregation and the cluster structure and, hence, substantially improves out-of-sample prediction accuracy relative to several benchmarks.

5. Simulated Experiments

In this section, we conduct computational experiments using simulated data. We focus on the DAC's predictive power and illustrate the improvement in prediction accuracy relative to several benchmarks in the non-asymptotic regime (i.e., when the data sample size is moderate or small). We consider two GLM settings: OLS (Section 5.1) and logistic regression (Section 5.3). The model's performance is evaluated using the out-of-sample R^2 for OLS and the area under the ROC curve (AUC) for logistic regression. We also undertake a comprehensive sensitivity analysis to investigate how the different parameters affect the model's performance. All the results presented in this section can be reproduced by using our available code and implementation details.[3]

5.1. Linear Regression

We assume that the data is generated from the following linear model:

$$Y_{i,j} = \sum_{l \in D_s} X^l_{i,j} \beta^s_l + \sum_{l \in D_n} X^l_{i,j} \beta^n_{i,l} + \sum_{l \in D_c} X^l_{i,j} \beta^c_{\mathcal{C}(i),l} + \epsilon_{i,j}, \qquad i = 1, \ldots, n \text{ and } j = 1, \ldots, m,$$

[3] `https://github.com/DACPublicator/DAC_Publication`

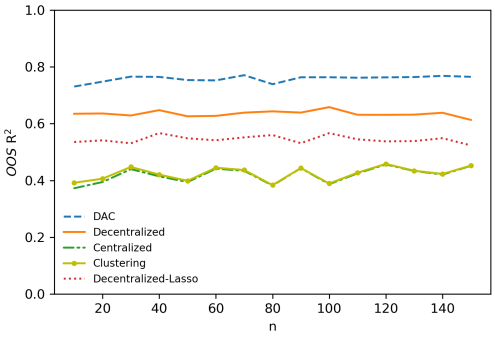
Table 2 Parameters used in Section 5.1.

Parameter	Range of values
Number of items (n)	$[10, 150]$
Number of features (d)	$[5, 15]$
Number of observations (m)	$[10, 50]$
Variance of the noise (σ^2)	$[0.25, 2]$
Department-level probability (p)	$[1/6, 2/3]$ or $[1/6, 1/3]$
Cluster-level probability (q)	$[1/6, 2/3]$ or $[1/6, 1/3]$

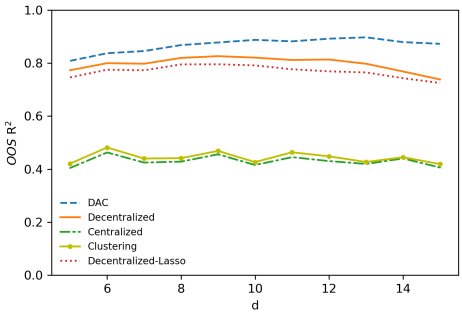
where $\epsilon_{i,j} \sim N(0, \sigma^2)$ are i.i.d. random variables. Each feature, $X^l_{i,j}$, is generated randomly from a uniform $[0, 1]$ distribution, and each β coefficient is obtained from a uniform $[-5, 5]$ distribution. We fix the number of clusters $k = 2$ and vary the parameters $\{n, d, m, \sigma^2, p, q\}$ one at a time (we observed similar results for alternative values of k). The definition and range of values for these parameters are reported in Table 2. The parameters p and q represent the probability that a given feature is modeled at the department and cluster levels, respectively (different features are drawn independently). Thus, the probability that a feature is at the item level is $(1 - p - q)$. It is important to note that the DAC's implementation admits three hyper-parameters (θ, R_U, and R_L) in addition to the number of clusters k. These three parameters represent the strictness of our algorithm in determining whether a feature should be aggregated. Specifically, θ is the p-value cutoff for statistical significance and is usually set at 0.05 or 0.01. The parameters R_U and R_L represent thresholds for the ratio of non-rejected hypotheses. For example, suppose that the percentage of non-rejected hypotheses for feature j is $R_j = 0.3$ (i.e., 30% of the items have statistically close estimated coefficients). Then, we label feature j as a department-level feature if $R_j > R_U$ and as a SKU-level feature if $R_j < R_L$. For any intermediate value $R_j \in [R_L, R_U]$, we will label feature j as a cluster-level feature. A good way to set the parameters R_L and R_U is by performing a cross-validation procedure (for more details, see Section 6). The parameters θ, R_U, and R_L provide flexibility in the tolerance level of the algorithm. In this section, we fine-tune these parameters by testing a grid of values. When implementing our algorithm with real data (Section 6), we will carefully set their values using a rigorous cross-validation procedure.

To test the performance of the DAC algorithm, we consider the four following benchmarks: decentralized, decentralized-Lasso, centralized, and clustering. For each problem instance (i.e., a specific combination of $\{n, d, m, \sigma, p, q\}$), we generate 100 independent trials (datasets) and use 67% as training and 33% as testing. We then report the average out-of-sample R^2 across all items and observations. Below is a description of the methods we consider:

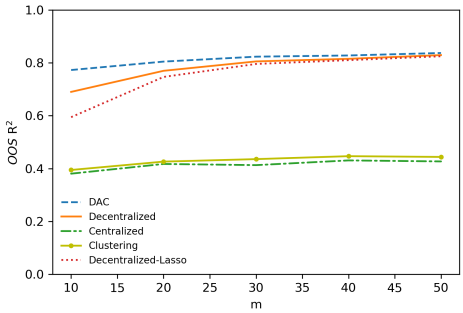
1. **DAC**: We implement our algorithm with $\theta = 0.05$, $R_U = 0.9$, and $R_L = 0.6$.
2. **Decentralized**: We estimate a simple OLS model for each item separately (i.e., n models).
3. **Decentralized-Lasso**: Same as the decentralized method, but we add an L_1 regularization term to each OLS model (we obtained similar results when using a regularization based on Ridge regression or Elasticnet).



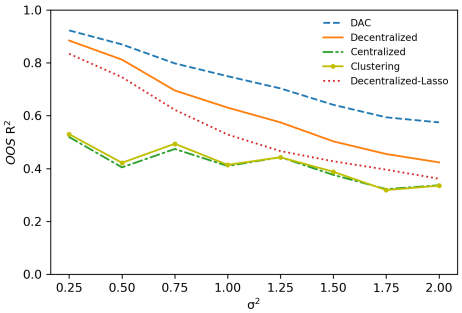
(a) Varying the number of items



(b) Varying the number of features



(c) Varying the number of observations



(d) Varying the noise magnitude

Figure 3 Comparison of prediction models (for linear regression).

4. **Centralized**: This is a naïve OLS model where we assume that for each feature, all the items have the same coefficient.

5. **Clustering**: We first cluster the items using k-means based on their features. We then fit an OLS model for each cluster.

As we can see from Figure 3, our algorithm outperforms all the benchmarks in all settings, in terms of out-of-sample R^2. Similar results were observed for the mean squared error (MSE). As expected, as we increase the number of observations, the prediction accuracy of the DAC algorithm quickly converges to 1. We find that the convergence is faster relative to the decentralized OLS method (with or without regularization), hence highlighting the benefit of the DAC algorithm under limited data availability. This clearly demonstrates the power of data aggregation and clustering present in our algorithm. Interestingly, the performance is not greatly affected by the number of items or by the number of features (indeed, the performance of each method depends on the proportion of the different feature types rather than the absolute number of features). Finally, as expected, a higher σ^2 negatively impacts the prediction accuracy of all methods, as it makes identifying the structure more challenging. Still, our proposed algorithm has a substantial advantage

Table 3 Performance statistics over 100 independent trials (each value is the out-of-sample R^2). Parameters: $m = 20$, $d = 8$, $n = 100$, $\sigma^2 = 1$, $p = 2/3$, $q = 1/6$.

Method	Min	Max	Mean	Std
DAC	0.691	0.943	0.876	0.038
Decentralized	0.564	0.924	0.825	0.061
Centralized	0.025	0.907	0.403	0.199
Clustering	0.056	0.907	0.405	0.197
Decentralized-Lasso	0.48	0.909	0.799	0.071

Table 4 Performance across items (each value is the out-of-sample MSE). Same parameters as Table 3.

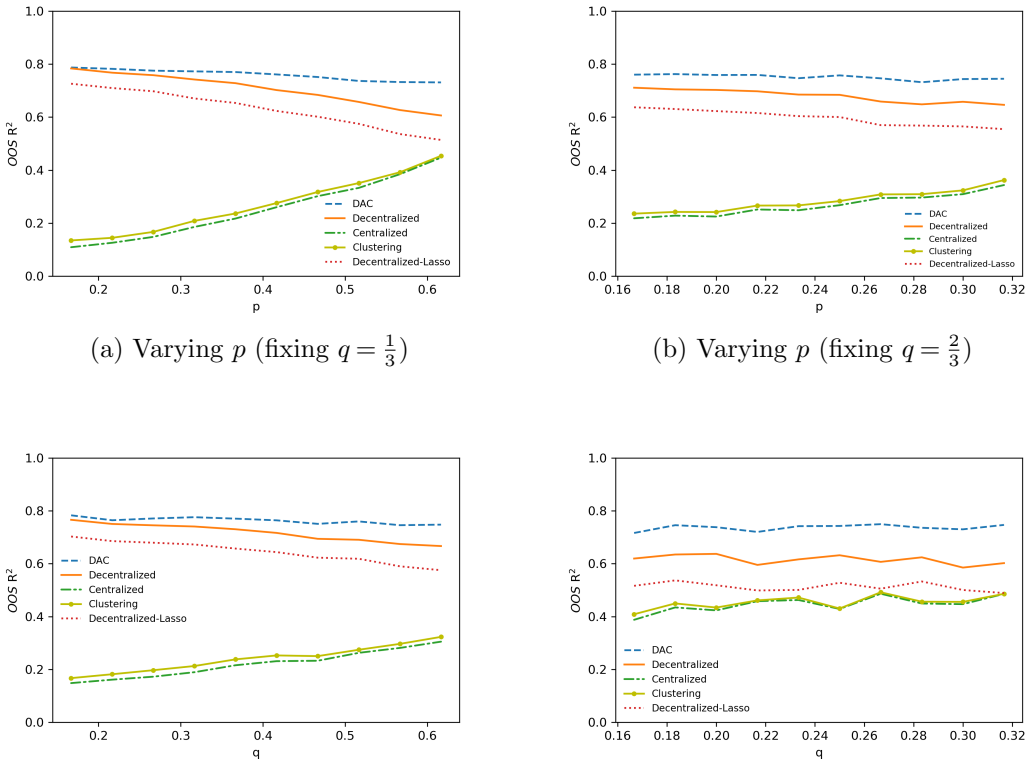
Method	Min	Max	Mean	Std
DAC	0.216	2.61	1.041	0.495
Decentralized	0.304	3.515	1.542	0.653
Centralized	0.386	13.25	3.725	3.02
Clustering	0.273	13.69	3.69	2.983
Decentralized-Lasso	0.285	6.728	1.784	1.07

relative to the four benchmarks. To complement the results of Figure 3, we report some statistics on the results of the different methods. In Table 3, we report the performance statistics (measured by the out-of-sample R^2) over the 100 independent trials. In Table 4, we report the performance statistics across $n = 100$ different items (in this case, the per-item performance is captured by the MSE). As we can see from both tables, the DAC algorithm outperforms all the benchmarks for all metrics. In addition, the standard deviation—both across instances and across items—reduces significantly, hence suggesting that our proposed algorithm also decreases the variability of the performance.

Figure 4 presents the performance of the methods when we vary the structure probability of the features in terms of aggregation level. When a large proportion of the features are at the department level (i.e., p is getting close to 1), all five methods perform well (the DAC algorithm still performs best in all cases). However, for instances where the structure is more diverse, our algorithm significantly outperforms the four benchmarks. Specifically, in all cases, the DAC algorithm leverages the structure of the problem by aggregating the data, ultimately yielding a higher prediction accuracy.

5.2. Two Types of Items

In this section, we consider an interesting setting where a subset of the items have limited data (referred to as 'new items'), whereas other items have abundant data (referred to as 'old items'). Our goal is to showcase that our proposed DAC algorithm can leverage the data from the old items to improve the prediction accuracy for *both* types of items. These computational experiments complement the analytical result derived in Proposition 4 by considering a non-asymptotic regime with more than two items. Specifically, we consider a setting with $n = 20$ items, $d = 5$ features, $\sigma^2 = 1$, $p = 2/3$, $q = 1/6$, $m_{new} = 10$, and $m_{old} = 30$. It thus corresponds to the situation where the



(a) Varying p (fixing $q = \frac{1}{3}$)

(b) Varying p (fixing $q = \frac{2}{3}$)

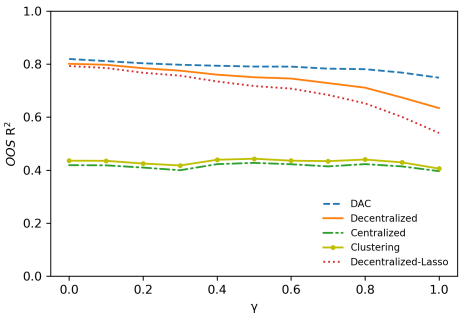
(c) Varying q (fixing $p = \frac{1}{3}$)

(d) Varying q (fixing $p = \frac{2}{3}$)

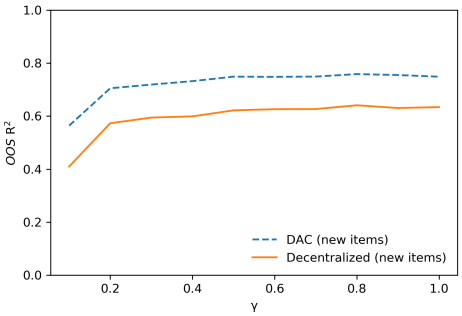
Figure 4 Comparison of prediction models (for linear regression).

old items have three times more observations than the new items. We then vary the proportion of new items, captured by the parameter $0 \leq \gamma \leq 1$. When $\gamma = 0$, it corresponds to the setting where all the items have abundant data (i.e., they all have $m_{old} = 30$ observations). As before, we consider 100 independent trials and use the same fine-tuned values of θ, R_U, and R_L.

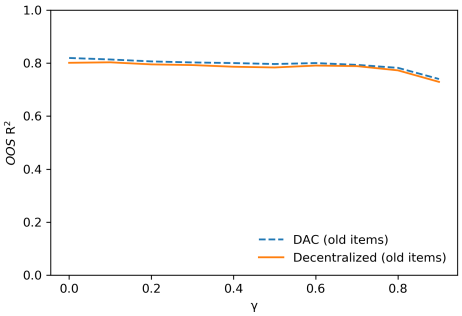
The results are presented in Figure 5. In the top panel, we show the average out-of-sample R^2 across all items as a function of γ. As before, the DAC algorithm consistently outperforms all four benchmarks. As expected, the benefit of the proposed algorithm relative to the decentralized OLS method increases as γ increases. In the bottom two panels, we compute the out-of-sample R^2 separately for new items (panel b) and old items (panel c) while focusing on the comparison between DAC and decentralized methods. The results readily confirm both insights drawn from Proposition 4: (i) the DAC algorithm improves the prediction accuracy for both types of items, and (ii) the improvement is more substantial for the items with limited data.



(a) All items



(b) New items only



(c) Old items only

Figure 5 Comparison of prediction models for a setting with two types of items (for linear regression).

5.3. Logistic Regression

In this section, we present computational experiments for a classification problem in which the data-generating process is the logistic regression model, that is, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$,

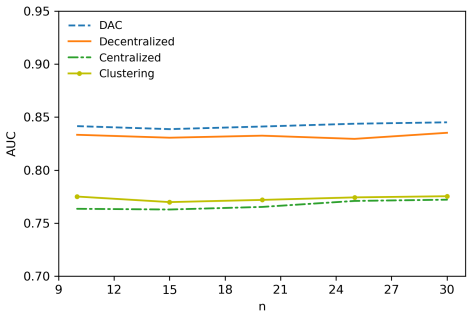
$$Y_{i,j} \sim Bernoulli\left(\mu_{i,j}\right), \qquad \mu_{i,j} = \text{logit}\left(\sum_{l \in D_s} X^l_{i,j}\beta^s_l + \sum_{l \in D_n} X^l_{i,j}\beta^n_{i,l} + \sum_{l \in D_c} X^l_{i,j}\beta^c_{\mathcal{C}(i),l}\right),$$

where $\text{logit(z)} := \frac{\exp(z)}{1+\exp(z)}$. We consider a similar setting as in Section 5.1 and use the same values of k, θ, R_U, and R_L as before. We first generate the data matrix X from a uniform $[0, 1]$ distribution and the β coefficients from a uniform $[-5, 5]$ distribution. The outcome variable Y is then generated based on a Bernoulli distribution with parameter $\mu = \text{logit}(X\beta)$. As in the OLS case, we vary one parameter at a time. The parameters' value ranges are summarized in Table 5.

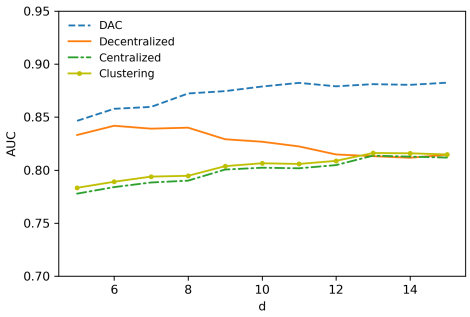
Table 5 Parameters used in Section 5.3.

Parameter	Range of values
Number of items (n)	$[10, 30]$
Number of features (d)	$[5, 15]$
Number of observations (m)	$[40, 100]$

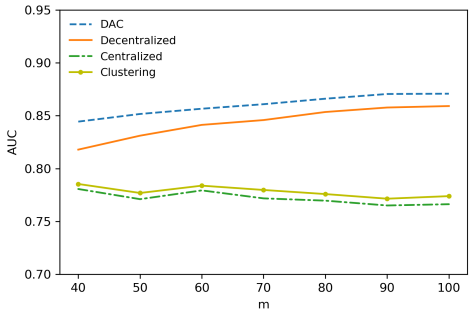
Following several prior studies on binary classification problems, we use the AUC as the metric to evaluate the performance of the different models. AUC is defined as the area under the receiver



(a) Varying the number of items



(b) Varying the number of features



(c) Varying the number of observations

Figure 6 Comparison of prediction models (for logistic regression).

operating characteristic (ROC) curve (see, e.g., Bradley 1997). It can be interpreted as the probability that a prediction model is correctly ranking a random positive outcome higher than a random negative outcome. We compare our DAC algorithm relative to three benchmarks: decentralized, centralized, and clustering (definitions are similar to Section 5.1).[4] For each instance, we generate 100 independent trials and report the average out-of-sample AUC scores.

As we can see from Figure 6, our method outperforms the benchmarks in all cases. Regardless of how we vary $\{n, m, d\}$, the DAC algorithm outperforms the three other methods in terms of prediction accuracy. Furthermore, a similar result as in Figure 4 was also observed for the logistic regression model (the details are omitted to avoid repetition).

To summarize, our simulation studies demonstrate a substantial and robust performance improvement for our proposed algorithm relative to several benchmarks (which are commonly used in practice and in the literature), even if the sample size is limited. Various other benchmarks will be considered in the next section, where the prediction algorithms are implemented. For both regression and classification problems, the DAC efficiently aggregates data and identifies the cluster

[4] Since estimating a decentralized model with L_1 regularization is computationally prohibitive for the logistic regression setting, we only show the performance of the decentralized model without regularization.

structure of the items, thus ultimately improving the prediction accuracy. In the next section, we apply the DAC algorithm using retail data to showcase its benefits in a practical setting.

6. Applying the DAC Algorithm to Retail Data

In this section, we apply the DAC algorithm to a retail dataset from a large global retailer (we cannot reveal the name of the retailer due to a non-disclosure agreement). We first provide a detailed description of the data and then test the prediction performance of the DAC algorithm relative to a broad range of benchmarks (we consider a total of 15 commonly used benchmarks). Finally, based on our computational findings, we draw managerial insights that can help retailers infer which features should be aggregated in practice.

6.1. Data

We have access to the retailer's online sales data. The dataset includes the weekly sales information of three departments between November 2013 and October 2016. A typical department comprises 100–150 SKUs. In addition to the weekly sales information, the dataset includes the weekly price, a promotion indicator (i.e., whether an item was promoted), the vendor, and the color of the SKU. Table 6 summarizes the specifics of each department. The size corresponds to the number of items in each department, and the numbers in parentheses are the standard deviations.

Table 6 Summary statistics of the data from each department.

Dept	Size	Observations	Weekly sales	Price	Promotion frequency	Discount rate
1	147	19,064	108.11 (377.45)	42.92 (38.80)	31.3%	4.2%
2	134	20,826	254.23 (517.07)	8.94 (8.67)	7.7%	6.4%
3	125	14,457	68.68 (691.55)	97.61 (67.12)	8.5%	1.5%

As we can see from Table 6, each department has a large number of observations (here, the observations are at the SKU-week level). There is also a great variation in terms of weekly sales, prices, promotion frequency, and discount rates across the different departments. Table 7 provides a brief description of the different fields in our dataset. The effective weekly price is computed as the total weekly revenue divided by the total weekly sales. Functionality is a segmentation hierarchy used by the firm to classify several SKUs from the same department into sub-categories.

Based on the features available in our data, we consider the following model specification:

$$\begin{aligned} Y_{i,t} = \beta^i_{\text{Trend}} \cdot T_{i,t} + \beta^i_0 \cdot p_{i,t} + \beta^i_1 \cdot \text{PromoFlag}_{i,t} + \beta^i_2 \cdot \text{Fatigue}_{i,t} + \beta^i_3 \cdot \text{Seasonality}_{i,t} + \\ + \beta^i_4 \cdot \text{Functionality}_{i,t} + \beta^i_5 \cdot \text{Color}_{i,t} + \beta^i_6 \cdot \text{Vendor}_{i,t} + \epsilon_{i,t}. \end{aligned} \tag{6}$$

Equation (6) includes the following features: $Y_{i,t}$ is the total weekly sales of item i in week t (our dependent variable), $T_{i,t}$ is the trend variable of item i (we normalize the year so that $T_i =$

Table 7 Fields in our dataset (observations are aggregated at the SKU-week level).

Fields	Description
SKU ID	Unique SKU ID
Week	Week index
Year	Year index
Units	Total weekly sales of a specific SKU
Price	Effective weekly price of a specific SKU
PromoFlag	Whether there was a promotion during that week
Functionality	Class index of a specific SKU
Color	Color of a specific SKU
Vendor	Vendor of a specific SKU

$0, 1, 2, 3$), $p_{i,t}$ is the effective price of item i in week t, $\text{PromoFlag}_{i,t}$ is a binary variable indicating whether there is a promotion for item i in week t, $\text{Fatigue}_{i,t}$ is the number of weeks since the most recent promotion for item i (if there is no previous promotion, $\text{Fatigue}_{i,t} = 0$; this feature allows us to capture the post-promotion dip effect), Seasonality is a categorical variable that measures the weekly or monthly effect on sales (we use one-hot encoding), and $\epsilon_{i,t}$ is an additive unobserved error. The remaining three variables are categorical variables indicating the web class index (functionality), color, and vendor of the SKU, respectively.

6.2. Prediction Performance

6.2.1. Benchmarks. As in Section 5, we compare the performance of the DAC algorithm relative to the same four benchmarks (decentralized, decentralized-Lasso, centralized, and clustering). In this section, for our analysis with real data, we also consider the following additional benchmarks: decentralized-elasticnet, DBSCAN clustering, OPTICS clustering, centralized-ISI (item-specific intercepts), centralized-K-means CFE (cluster-fixed effects), centralized-DBSCAN CFE, centralized-OPTICS CFE, decision tree, random forest, gradient boosted tree, and neural network.[5] Note that we consider three different clustering methods as benchmarks (k-means, DBSCAN, and OPTICS), as well as machine-learning based methods. Finally, we test adding fixed effects to the decentralized model under several configurations. We thus compare the DAC algorithm's performance to a total of 15 benchmarks, which are commonly used in both academia and practice.

It is important to mention that when implementing our DAC algorithm, we need to slightly adapt the clustering step of the algorithm. To ensure that we output a single cluster structure, we first collect the estimated coefficients from all cluster-level features and then fit a multi-dimensional k-means model. To avoid over-fitting, one can also include a L_1-regularization term in the aggregated estimation (Step 6 of Algorithm 1).

6.2.2. DAC Implementation. Since the DAC algorithm has four hyper-parameters (k, θ, R_U, R_L), we use an extensive cross-validation procedure to fine-tune these parameters and

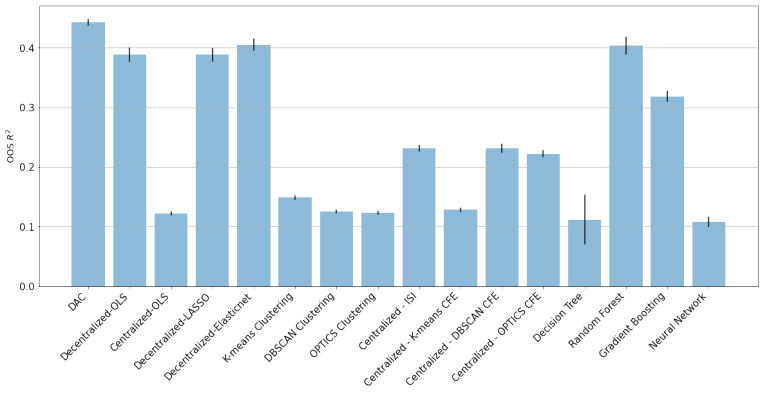
[5] The implementation details of the aforementioned methods are quite generic and are available upon request.

select the best model. For each department, we first randomly split the data into training (70%) and testing (30%) sets. We assume that each design parameter lies within a pre-specified range: $k \in \{3, 4, \ldots, 10\}$, $\theta \in \{0.01, 0.05, 0.1, 0.5\}$, $R_U \in \{0.7, 0.8, 0.9\}$, and $R_L \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ (we add the option $\theta = 0.5$ to include the decentralized method as a special case). For each combination of hyper-parameters, we perform a five-fold cross-validation by fitting the model on 80% of the training data and compute the R^2 based on the remaining 20% of the data. This procedure is repeated five times for each parameter combination, and we compute the average R^2 over the five folds. We next select the best model based on the average cross-validation performance. Finally, the out-of-sample R^2 is computed on the testing set. Furthermore, since the train/cross-validation/test split is done randomly, we conduct 100 independent trials and report the mean and the 95%-confidence interval of the out-of-sample R^2.

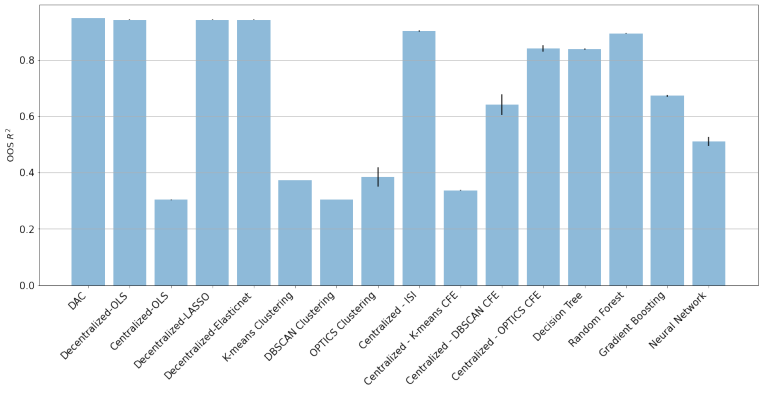
Our computational environment relies on the resources of Compute Canada and, more precisely, on the Volta generation Nvidia V100-SXM2-16Go node accessible on the Beluga computing network, with 6Go of memory (V100 offers the performance of up to 100 CPUs in a single GPU).

6.2.3. Results. We present the results for the out-of-sample R^2 using a five-fold cross validation in Figure 7. As expected, we obtain similar results for the MSE and the mean absolute percentage error (the details are omitted for conciseness). In Figure 7, each bar represents the average out-of-sample performance, and the length of the vertical line corresponds to the 95%-confidence interval across 100 trials. For all three departments, the DAC algorithm not only achieves a better average prediction performance but also often has a smaller variance. This shows that our algorithm is robust to different train/test splits, which is very desirable in practice. In addition, DAC outperforms all 15 benchmarks regardless of data quality. More precisely, Department 2 seems to have high-quality data and predictability power, whereas the data for Department 3 seems to be of lower quality (the number of observations per SKU is the smallest for Department 3, and data variability is high). Irrespective of the data quality, the DAC algorithm yields a clear improvement in prediction accuracy relative to all the benchmarks we considered. For completeness, we also recorded the in-sample R^2 values and consistently observed that the DAC algorithm reduces the amount of over-fitting relative to the other methods.

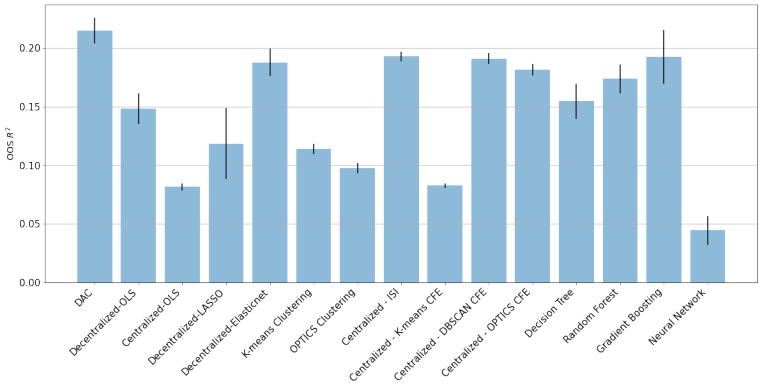
6.2.4. Time-Based Split Results. So far, we considered testing the different methods by using a random-based split. This allows us to perform a cross validation and, ultimately, compute confidence intervals in order to make statistical claims on the performance comparisons. An alternative way is to use a time-based split. Specifically, we can split the data by using the first $Y\%$ weeks for training (e.g., $Y = 70$) and the remaining $1 - Y\%$ for testing. This alternative data splitting rule is attractive in terms of prediction applicability but lacks statistical support in terms



(a) Department 1



(b) Department 2



(c) Department 3

Figure 7 **Performance comparison using real data (metric: out-of-sample R^2, random split).**

of comparing the different methods. When opting for a time-based split, two options are possible: an item-based split or an absolute split. In an item-based split, we divide the data into training and testing sets for each item separately. In an absolute split, we look at the total number of weeks in the dataset and then divide the data for all the items using the same week threshold. These two options differ when different items are introduced to the assortment at different times. In Figure 8, we present the results for Department 2 (the highest performing department) under

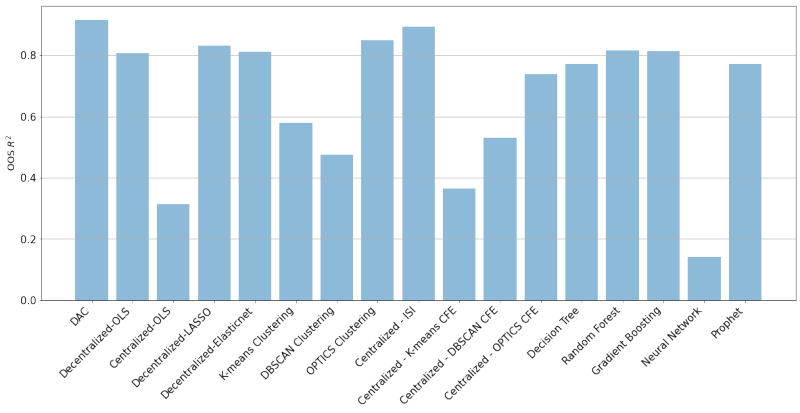


Figure 8 **Comparison using real data for Department 2 (metric: out-of-sample R^2, time-based split).**

Table 8 **Number of estimated coefficients.**

Department	Decentralized	DAC
1	9,408	6,095
2	6,298	4,977
3	3,000	892

an item-based split using 70% for training and 30% for testing. We obtained similar results for other split ratios and when using an absolute split. When using a time-based split, we can also consider time-series methods. Specifically, we include the Prophet method as an additional benchmark (Taylor and Letham 2018), which is considered among the state-of-the-art time-series approaches for demand prediction. Overall, as in the random-based split, the DAC algorithm outperforms all the benchmarks.

In summary, our results based on real data from three retail departments strongly suggest that the DAC algorithm has superior performance in terms of prediction accuracy. In addition to outperforming 15 benchmarks, our method reduces over-fitting and provides a great interpretability advantage. Specifically, it can help retailers identify the correct level of data aggregation for the different features and uncover the underlying cluster structure. Unlike other methods, the model output yields meaningful managerial insights, as we discuss next.

6.3. Managerial Insights

So far, we focused on the prediction performance of our proposed method. We then apply the DAC algorithm to our dataset and examine the estimation output. Our goal is to draw managerial insights into the hierarchical structure of the features. Next, we summarize our findings.

- The DAC algorithm can significantly reduce the model dimension. In Table 8, we report the number of estimated coefficients for the decentralized and DAC approaches across all three departments. Depending on the department, the number of estimated coefficients is reduced by 20% to 70%. The results in Table 8 confirm that shared coefficients do occur in practice and that data aggregation can play an important role in correctly identifying the aggregation structure.

• Practitioners often argue that seasonality features should be aggregated at the department level for demand prediction (e.g., Cohen et al. 2017, Vakhutinsky et al. 2018). Using our retail dataset, we discover that this is indeed the case. If we model seasonality at the month level (i.e., we use 11 dummy variables for each calendar month), we find that for two departments, all 11 variables should indeed be estimated at the department level (whereas for the third department, the same is true for 7 out of 11 variables). If we instead model seasonality at the week level (i.e., we use 51 dummy variables for each week of the year), we find that 48 out of 51 variables should be estimated at the department level for two departments (and 18 out 51 for the third department). Thus, our findings validate and refine a well-known insight in practice.

• The price feature is unarguably one of the most important features for demand prediction in retail. According to our estimation results, for all three departments, we obtain a distinct coefficient for the price feature, implying that the price coefficient should be estimated at the SKU level. This typically holds for departments with a heterogenous item collection.

• We find that the fatigue and promotion features should be estimated at the department level for all three departments (except the fatigue feature for one department). This is an interesting insight that can guide retailers when deciding their promotion strategy.

• We also find that all vendor and color dummy variables should be estimated at the SKU level. This is unsurprising given that most vendor-color combinations are unique for a specific SKU.

• The functionality dummy variables have different aggregation levels and cluster structures. Interestingly, most cluster-level features come from functionality. One possible explanation is that the functionality feature is obtained based on the hierarchy structure used by the company. Thus, SKUs with similar characteristics are usually labeled under the same functionality, making the cluster structure more prominent for functionality features. Retailers can use such results to potentially revise and improve their hierarchical structure and product segmentation.

7. Conclusion

Demand prediction (or sales forecasting) is an important task faced by most retailers. Improving prediction accuracy and drawing insights on data aggregation can significantly impact retailers' decisions and profits. When designing and estimating predictive models, retailers need to decide the aggregation level of each feature (e.g., seasonality and price). Some features may be estimated at the SKU level, others at the department level, and the rest at a cluster level. Traditionally, this problem was addressed by trial-and-error or by relying on past experience. It is common to see data scientists testing a multitude of model specifications until they find the best aggregation level for each feature. Such an ad-hoc approach can be tedious and is not scalable for cases with a large number of features. The goal of this paper is to develop an efficient method to simultaneously

determine (i) the correct aggregation level of each feature, (ii) the underlying cluster structure, and (iii) the estimated coefficients.

We propose a method referred to as the data aggregation with clustering (DAC) algorithm. The DAC approach can determine the correct aggregation level and identify the cluster structure of the items. This method is tractable even when data dimensionality is high, and it can significantly improve the efficiency in estimating the model coefficients. We first derive several analytical results to demonstrate the benefits of our proposed method. Specifically, we show that the DAC algorithm yields a consistent estimate, along with improved asymptotic properties relative to the decentralized method. We then go beyond the theory and implement the DAC algorithm using a large retail dataset. In all our computational tests, we observe that the proposed method significantly improves prediction accuracy relative to a multitude of benchmarks. Finally, we convey that our method can help retailers uncover useful insights from their data.

Acknowledgments

We would like to thank Paul-Emile Gras and Arthur Pentecoste who helped us conduct the computations presented in Sections 5 and 6. We would also like to thank the retail partner for sharing data and Compute Canada for allowing us to use their computing resources.

References

Baardman L, Levin I, Perakis G, Singhvi D (2017) Leveraging comparables for new product sales forecasting, available at SSRN 3086237.

Bernstein F, Modaresi S, Sauré D (2019) A dynamic clustering approach to data-driven assortment personalization. *Management Science* 65(5):2095–2115.

Bertsimas D, Kallus N (2020) From predictive to prescriptive analytics, management Science.

Bradley AP (1997) The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition* 30(7):1145–1159.

Caro F, Gallien J (2010) Inventory management of a fast-fashion retail network. *Operations Research* 58(2):257–273.

Cohen MA, Lee HL (2020) Designing the right global supply chain network. *Manufacturing & Service Operations Management* 22(1):15–24.

Cohen MC, Kalas JJ, Perakis G (2020) Promotion optimization for multiple items in supermarkets, forthcoming at Management Science.

Cohen MC, Leung NHZ, Panchamgam K, Perakis G, Smith A (2017) The impact of linear optimization on promotion planning. *Operations Research* 65(2):446–468.

Cooper LG, Baron P, Levy W, Swisher M, Gogos P (1999) Promocast$^{\text{TM}}$: A new forecasting method for promotion planning. *Marketing Science* 18(3):301–316.

Donti P, Amos B, Kolter JZ (2017) Task-based end-to-end model learning in stochastic optimization. *Advances in Neural Information Processing Systems*, 5484–5494.

Elmachtoub AN, Grigas P (2017) Smart" predict, then optimize", arXiv preprint arXiv:1710.08005.

Fahrmeir L, Kaufmann H, et al. (1985) Consistency and asymptotic normality of the maximum likelihood estimator in generalized linear models. *The Annals of Statistics* 13(1):342–368.

Feng Q, Shanthikumar JG (2018) How research in production and operations management may evolve in the era of big data. *Production and Operations Management* 27(9):1670–1684.

Fildes R, Goodwin P, Önkal D (2019) Use and misuse of information in supply chain forecasting of promotion effects. *International Journal of Forecasting* 35(1):144–156.

Greene WH (2003) *Econometric analysis* (Pearson Education India).

Hu K, Acimovic J, Erize F, Thomas DJ, Van Mieghem JA (2017) Forecasting product life cycle curves: Practical approach and empirical analysis, manufacturing & Service Operations Management.

Huang T, Fildes R, Soopramanien D (2014) The value of competitive information in forecasting fmcg retail product sales and the variable selection problem. *EJOR* 237(2):738–748.

Huang T, Fildes R, Soopramanien D (2019) Forecasting retailer product sales in the presence of structural change, european Journal of Operational Research.

Jagabathula S, Subramanian L, Venkataraman A (2018) A model-based embedding technique for segmenting customers. *Operations Research* 66(5):1247–1267.

Kesavan S, Gaur V, Raman A (2010) Do inventory and gross margin data improve sales forecasts for us public retailers? *Management Science* 56(9):1519–1533.

Kök AG, Fisher ML (2007) Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research* 55(6):1001–1021.

Laurent B, Massart P (2000) Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics* 1302–1338.

Lin M, Lucas Jr HC, Shmueli G (2013) Research commentary—too big to fail: large samples and the p-value problem. *Information Systems Research* 24(4):906–917.

Liu S, He L, Max Shen ZJ (2020) On-time last-mile delivery: Order assignment with travel-time predictors, forthcoming at Management Science.

Macé S, Neslin SA (2004) The determinants of pre-and postpromotion dips in sales of frequently purchased goods. *Journal of Marketing Research* 41(3):339–350.

MacQueen J, et al. (1967) Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1(14), 281–297.

McCullagh P (2019) *Generalized linear models* (Routledge).

Taylor SJ, Letham B (2018) Forecasting at scale. *The American Statistician* 72(1):37–45.

Vakhutinsky A, Mihic K, Wu SM (2018) A prescriptive analytics approach to markdown pricing for an e-commerce retailer, URL `http://dx.doi.org/10.13140/RG.2.2.35292.69767`, working paper.

Van Heerde HJ, Leeflang PS, Wittink DR (2000) The estimation of pre-and postpromotion dips with store-level scanner data. *Journal of Marketing Research* 37(3):383–395.

Wang H, Song M (2011) Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal* 3(2):29.

Appendix A: Two Potential Methods

In this section, we elaborate on the key difficulties of two intuitive methods—*constrained MLE* and *iterative optimization*—in estimating Equation (1).

A.1. Constrained MLE

Constrained MLE adapts the standard MLE approach to account for the equality constraints $\beta_{i,l} = \beta_{j,l}$, for all i, j if l is an aggregated-level feature; and $\beta_{i,l} = \beta_{j,l}$, for all i, j in the same cluster if l is a cluster-level feature. This method, however, is challenging to implement. We next use a simple example to illustrate the main difficulty. Consider a linear regression model with two items (each with one feature) but the level of aggregation is unknown. The data matrix can be written as follows:

$$X = \begin{bmatrix} X_1^1 & 0 & X_1^1 \\ X_2^1 & 0 & X_2^1 \\ \dots & \dots & \dots \\ X_m^1 & 0 & X_m^1 \\ 0 & X_1^2 & X_1^2 \\ 0 & X_2^2 & X_2^2 \\ \dots & \dots & \dots \\ 0 & X_m^2 & X_m^2 \end{bmatrix}.$$

The data matrix X enumerates all possible column combinations (three in this case), which represent all possible options for feature aggregation. If the feature is at the SKU level, then only the first two columns will have non-zero coefficients. On the other hand, if the feature is at the department level, only the last column will have a non-zero coefficient. We denote the vector of parameters as $\beta = (\beta_1, \beta_2, \beta_{12})$. The aggregation level inference problem can be formulated as the following constrained MLE problem (for linear regression, MLE is equivalent to minimizing the squared error):

$$\begin{aligned} \min_{\beta} \quad & ||X\beta - y||^2 \\ \text{s.t.} \quad & \beta_1 \beta_{12} = 0, \\ & \beta_2 \beta_{12} = 0. \end{aligned}$$

Applying the Karush-Kuhn-Tucker (KKT) condition, we obtain the matrix-form optimality condition:

$$2X^T X\beta - 2X^T y + B^T \lambda = 0,$$

where

$$B = \begin{bmatrix} \beta_{12} & 0 & \beta_1 \\ 0 & \beta_{12} & \beta_2 \end{bmatrix}.$$

The complete KKT condition is then given by:

$$\begin{bmatrix} 2X^T X & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} 2X^T y \\ 0 \end{bmatrix}$$

which is a non-convex quadratic equation. Thus, the traditional approach to solve a constrained linear regression does not apply in this case. In addition, due to the non-convexity, it is not easy to solve the above optimization problem when the number of items or the number of features become large. For a non-linear GLM model, the constrained MLE approach is even more challenging. As a result, the constrained MLE does not seem to be an efficient method to solve our problem.

A.2. Iterative Optimization

The iterative optimization approach was proposed by Baardman et al. (2017). We use the same example as the constrained MLE (i.e., a linear model with only SKU- and department-level features) to illustrate the difficulty of applying the iterative optimization procedure. We formulate the estimation problem as the following optimization program:

$$\begin{aligned} \min_{\delta,\beta_n,\beta_0} \quad & \sum_{j=1}^{m}\sum_{i=1}^{n}\left(Y_{i,j}-\sum_{l=1}^{d}\left[X_{i,j}^{l}\beta_{i,l}\delta_l+X_{i,j}^{l}\beta_{0,l}(1-\delta_l)\right]\right)^2 \\ \text{s.t.} \quad & \beta_{i,l}\in\mathbb{R}^d \;\; i=1,2,\ldots,n,\;\; l=1,2,\ldots,d, \\ & \delta_l\in\{0,1\} \;\; l=1,2,\ldots,d, \end{aligned} \tag{7}$$

where i is the observation index, j the item index, and l the feature index. $\beta_0=(\beta_{0,1},\beta_{0,2},\ldots,\beta_{0,d})'$ (resp. $\beta_j=(\beta_{j,1},\beta_{j,2},\ldots,\beta_{j,d})'$) are the department-level (resp. SKU-level) coefficients for the features. This iterative optimization formulation is a quadratic mixed-integer program. Note that the model is simplified since it does not include cluster-level coefficients.

One can solve the optimization problem in (7) by using the following iterative algorithm:

1. Randomly initialize $\hat{\delta}^0\in\{0,1\}^d$ and solve for $\hat{\beta}^0=\{\hat{\beta}^0_{i,l}: i=1,2,\ldots,l=1,2,\ldots,d\}$ by fixing $\hat{\delta}=\hat{\delta}^0$.
2. Starting from iteration $t=1$, first solve for $\hat{\delta}^t$ by fixing $\beta=\hat{\beta}^{t-1}$; then solve for $\hat{\beta}^t$ by fixing $\delta=\hat{\delta}^t$.
3. Terminate Step 2 when $\hat{\delta}^t=\hat{\delta}^{t-1}$. The output is $(\hat{\beta}^t,\hat{\delta}^t)$.

The difficulty of the iterative optimization procedure is that without any prior information on β, the generation of coefficient vector is random. Then, if the generated coefficients are far from their true values, the procedure can converge to a local optimal solution and, hence, potentially yield a large estimation error. Furthermore, it is not clear how to incorporate the (unknown) cluster structure into the iterative optimization procedure. Thus, it is not guaranteed that the iterative optimization approach will lead to a consistent estimate of the feature coefficients, data aggregation levels, and cluster structure.

Appendix B: Proofs of Statements

B.1. Proof of Lemma 1

The proof follows from a standard result in statistics stating that the maximum-likelihood estimator (MLE) is consistent under some regularity conditions which are satisfied by a generalized linear model. See, for example, Fahrmeir et al. (1985) and McCullagh (2019).

B.2. Proof of Proposition 1

We analyze each aggregation level separately. The results build on Lemma 1 stating that all $\hat{b}_{i,j}$ are consistent.

- Case 1: Feature l is at the SKU level.

In this case, $b_{1,l}\neq b_{k,l}$ for all k in the decentralized model. As a result, based on Lin et al. (2013), for any $\eta>0$, we have that the p-value for the hypothesis $H_0: b_{1,l}=b_{k,l}$ satisfies that

$$\lim_{m\to\infty} p\text{-value} = \lim_{m\to\infty} Pr(|\hat{b}_{1,l}-\hat{b}_{k,l}|<\eta)=0. \tag{8}$$

Therefore, the p-value converges to 0 as $m \to \infty$. Hence, H_0 will be rejected for all k. Alternatively if $l \notin D_n$, there exists an item k', such that $b_{1,l} = b_{k',l}$. We have

$$\lim_{m\to\infty} p\text{-value} = \lim_{m\to\infty} \mathbb{P}(|\hat{b}_{1,l} - \hat{b}_{k',l}| < \eta) = 1. \tag{9}$$

Combining inequalities (8) and (9) implies that the probability that we misclassify feature l (in terms of whether feature l is at the SKU level) converges to 0 as $m \to \infty$, that is, the DAC algorithm consistently identifies whether feature l is at the SKU level.

- Case 2: Feature l is at the department level.

In this case, $b_{1,l} = b_{k,l}\ \forall k$ in the decentralized model. Again, from Lin et al. (2013), for any $\eta > 0$ we have

$$\lim_{m\to\infty} p\text{-value} = \lim_{m\to\infty} Pr(|\hat{b}_{1,j} - \hat{b}_{k,j}| < \eta) = 1. \tag{10}$$

Therefore, the p-value converges to 1 as $m \to \infty$. Alternatively if $l \notin D_s$, there exists an item k', such that $b_{1,l} \neq b_{k',l}$. We have

$$\lim_{m\to\infty} p\text{-value} = \lim_{m\to\infty} \mathbb{P}(|\hat{b}_{1,l} - \hat{b}_{k',l}| < \eta) = 0. \tag{11}$$

Combining inequalities (10) and (11) implies that the probability that we misclassify feature l (in terms of whether feature l is at the department level) converges to 0 as $m \to \infty$, that is, the DAC algorithm consistently identifies whether feature l is at the department level.

- Case 3: Feature l is at the cluster level.

As in the previous cases, we know that as $m \to \infty$, all the coefficients center around their true values with an arbitrarily high probability. Given the number of clusters k, the task is to partition $\{\hat{b}_{i,l} : i = 1, 2, , \cdots, n\}$ into k groups so that the sum of squared Euclidean distances to each group mean is minimized. In general, the high-dimensional k-means algorithm is NP-hard. However, for this specific one-dimensional k-means problem, there exists a dynamic programming algorithm, with a polynomial time complexity, which finds the optimal solution (Wang and Song 2011). This implies that the DAC algorithm consistently identifies whether feature l is at the cluster level and the correct cluster structure of the items. This concludes the proof. □

B.3. Proof of Proposition 2

Part (a). The result follows directly from Theorem 3 in Fahrmeir et al. (1985).

Part (b). Based on the decentralized estimator, $\hat{b}_i(m)$ $(i = 1, 2, ..., n)$, we construct the following new estimator $\hat{\theta}_{i,l}(m)$ for each item i and feature l:

$$\hat{\theta}_{i,l}(m) = \frac{1}{n_{i,l}} \sum_{i' \in \mathcal{C}(i,l)} \hat{b}_{i',l}(m).$$

From the consistency and asymptotic normality of $\hat{b}(m)$, we have $\hat{\theta}_{i,l}(m) \xrightarrow{p} \beta_{i,l},$ for each i and l; and

$$\lim_{m\to+\infty} m \cdot n_{i,l} \cdot \mathrm{Var}(\hat{\theta}_{i,l}(m)) = \frac{1}{n_{i,l}} \sum_{i' \in \mathcal{C}(i,l)} \kappa_{i',l}(m), \text{ for } i = 1, 2, \ldots, n,\ l = 1, 2, \ldots, d. \tag{12}$$

We next apply the Taylor expansion of $\nabla l_i(\cdot; m)$ arround the true parameter value β_i for each i:

$$\nabla l_i(\hat{b}_i(m); m) = \nabla l_i(\beta_i; m) + \nabla_2 l_i(\beta_i; m) \cdot (\hat{b}_i(m) - \beta_i) + o(|\hat{b}_i(m) - \beta_i|).$$

Since $\hat{b}_i(m)$ is the maximizer of $l_i(\cdot;m)$, $\nabla l_i(\hat{b}_i(m);m)=0$ and, thus,

$$\nabla l_i(\beta_i;m)+\nabla_2 l_i(\beta_i;m)\cdot(\hat{b}_i(m)-\beta_i)+o(|\hat{b}_i(m)-\beta_i|)=0, \text{ for each } i. \tag{13}$$

Analogously, we apply the Taylor expansion of $\nabla l(\cdot;m)$ around the true parameter value β:

$$\nabla l(\hat{\beta}(m);m)=\nabla l(\beta;m)+\nabla_2 l(\beta;m)\cdot(\hat{\beta}(m)-\beta)+o(|\hat{\beta}(m)-\beta|),$$

that is, $\sum_{i=1}^n \nabla l_i(\hat{\beta}_i(m);m)=\sum_{i=1}^n \nabla l_i(\beta;m)+\sum_{i=1}^n \nabla_2 l_i(\beta_i;m)\cdot(\hat{\beta}_i(m)-\beta_i)+o(|\hat{\beta}(m)-\beta|^2)$. Since $\hat{\beta}(m)$ is the maximizer of $l(\cdot;m)$ under the constraint that $\beta_{i,l}=\beta_{i',l}$ for all $i'\in\mathcal{C}(i,l)$, we have $\sum_{i'\in\mathcal{C}(i,l)}\nabla^l l_{i'}(\hat{\beta}_{i'}(m);m)=0$, where the operator ∇^l refers to the partial derivative with respect to feature l. Thus, for each item i, it follows that

$$\sum_{i'\in\mathcal{C}(i,l)}\nabla^l l_{i'}(\hat{\beta}_{i'}(m);m)=\sum_{i'\in\mathcal{C}(i,l)}\nabla^l l_{i'}(\beta_{i'};m)+\sum_{i'\in\mathcal{C}(i,l)}\nabla_2^l l_{i'}(\beta_{i'};m)\cdot(\hat{\beta}_{i'}(m)-\beta_{i'})+o(|\hat{\beta}(m)-\beta|)=0, \tag{14}$$

where ∇_2^l is the l-th row of the Hessian.

By Equation (13), we have for each i and each l,

$$\sum_{i'\in\mathcal{C}(i,l)}\nabla^l l_{i'}(\hat{b}_{i'}(m);m)=\sum_{i'\in\mathcal{C}(i,l)}\nabla^l l_{i'}(\beta_{i'};m)+\sum_{i'\in\mathcal{C}(i,l)}\nabla_2^l l_{i'}(\beta_{i'};m)\cdot(\hat{b}_{i'}(m)-\beta_{i'})+o(|\hat{b}(m)-\beta|)=0. \tag{15}$$

Note that $\hat{\beta}_{i',l}(m)=\hat{\beta}_{i,l}(m)$ for all $i'\in\mathcal{C}(i,l)$, so that there are $n_{i,l}=|\mathcal{C}(i,l)|$ coefficients identical to $\hat{\beta}_{i,l}(m)$. Plugging this identity into (14) and subtracting (15), we obtain the following:

$$|n_{i,l}\hat{\beta}_{i,l}(m)-\sum_{i'\in\mathcal{C}(i,l)}\hat{b}_{i,l}(m)|=o(m^{-\frac{1}{2}}), \text{ i.e., } |\hat{\beta}_{i,l}(m)-\hat{\theta}_{i,l}(m)|=o(m^{-\frac{1}{2}}),$$

where we used the facts that $|\hat{b}(m)-\beta|=O(m^{-\frac{1}{2}})$ and $|\hat{\beta}(m)-\beta|=O(m^{-\frac{1}{2}})$ (by applying the strong law of large numbers, namely, $\lim_{m\uparrow+\infty}\nabla_2^l l_i(\beta_i;m)=-\mathcal{I}_i^l(\beta_i)$ for any i and l), Therefore,

$$|\mathrm{Var}(\hat{\beta}_{i,l}(m))-\mathrm{Var}(\hat{\theta}_{i,l}(m))|=o(m^{-1}).$$

Using Equation (12) and the Cauchy-Schwartz inequality, we have

$$\lim_{m\to+\infty} m\cdot n_{i,l}\mathrm{Var}(\hat{\beta}_{i,l}(m))=\frac{1}{n_{i,l}}\sum_{i'\in\mathcal{C}(i,l)}\kappa_{i',l}, \text{ for } i=1,2,...,n,\ l=1,2,...,d.$$

This concludes the proof of Proposition 2. □

B.4. Proof of Proposition 3

First, for the aggregated model, we have

$$||X\beta-X\hat{\beta}||_2^2=||X\beta-X\beta-X(X'X)^{-1}X'\epsilon||_2^2=||P\epsilon||_2^2,$$

where $P=X(X'X)^{-1}X'$ is an idempotent matrix (i.e., a matrix which, when multiplied by itself, yields itself). Since $\epsilon\sim N(0,\sigma^2 I)$, we can write,

$$\frac{||X\beta-X\hat{\beta}||_2^2}{\sigma^2}=\left(\frac{\epsilon'}{\sigma}\right)P\left(\frac{\epsilon}{\sigma}\right),$$

where ϵ/σ is a standard normal vector. Based on Section B11.4 in Greene (2003), the above quantity follows a χ^2 distribution with degrees of freedom equal to rank(P). By the commutativity property of the trace operator, we have $tr(P) = tr(X(X'X)^{-1}X') = tr(X'X(X'X)^{-1}) = d_x$, which is the column rank of matrix X. If X has full rank, then d_x represents the number of features under the aggregated model.

Based on Lemma 1 in Laurent and Massart (2000), we can use the tail bound of χ^2 distribution on the mean squared errors. For any $\gamma > 0$, we have the following high-probability upper bound:

$$\begin{aligned} &\mathbb{P}\left(\frac{||X\beta - X\hat{\beta}||_2^2}{\sigma^2} - d_x \geq 2\sqrt{\gamma d_x} + 2\gamma\right) \leq \exp(-\gamma), \\ \implies &\mathbb{P}\left(\frac{||X\beta - X\hat{\beta}||_2^2}{n \times m} \leq \frac{\sigma^2\left(2\sqrt{\gamma d_x} + 2\gamma + d_x\right)}{n \times m}\right) \geq 1 - \exp(-\gamma). \end{aligned} \tag{16}$$

On the other hand, if we use a simple OLS for each item, we obtain n terms of squared errors $||X_i b_i - X_i\hat{b}_i||_2^2$. Each term is a χ^2 distributed variable with degrees of freedom equal to d. Thus, when computing the MSE for the decentralized model, we obtain the following:

$$\mathbb{P}\left(\frac{\sum_{i=1}^n ||X_i b_i - X_i\hat{b}_i||_2^2}{n \times m} \leq \frac{\sigma^2\left(2\sqrt{\gamma\,(nd)} + 2\gamma + nd\right)}{n \times m}\right) \geq 1 - \exp(-\gamma). \tag{17}$$

Note that $d_x \in [d, nd]$. Therefore, unless all the features are at the SKU level, we have $d_x < nd$ and, thus, the bound in (16) is tighter than the bound in (17). Consequently, we can achieve a smaller MSE for the aggregated model relative to the decentralized model under the same level of confidence, $\exp(-\gamma)$.

In addition, we can provide a high-probability lower bound for the decentralized model. As shown in Laurent and Massart (2000), we have

$$\mathbb{P}\left(\frac{\sum_{i=1}^n ||X_i b_i - X_i\hat{b}_i||_2^2}{n \times m} \geq \frac{\sigma^2\left(nd - 2\sqrt{\gamma\,(nd)}\right)}{n \times m}\right) \geq 1 - \exp(-\gamma). \tag{18}$$

If we compare the upper bound in (16) to the lower bound in (18), we can solve for the sufficient condition under which the aggregated model outperforms the decentralized model with high probability:

$$\frac{\sigma^2\left(2\sqrt{\gamma d_x} + 2\gamma + d_x\right)}{n \times m} \leq \frac{\sigma^2\left(nd - 2\sqrt{\gamma\,(nd)}\right)}{n \times m} \implies d_x^2 - (2A + 4\gamma)d_x + A^2 \geq 0,$$

where $A = nd - 2\sqrt{\gamma nd} - 2\gamma$. As a result, the above inequality holds when

$$d_x \leq nd - 2\left(\sqrt{\gamma nd} + \sqrt{\gamma nd - 2\gamma\sqrt{nd} - \gamma^2}\right)$$

and this concludes the proof of Proposition 3. □

B.5. Proof of Proposition 4

As a first step, we adopt the bias-variance decomposition (e.g., Equation (7.9) of Hastie et al. 2009) to evaluate the generalization error. Specifically, for any estimation algorithm π, we have

$$\begin{aligned}\mathsf{e}_i(\pi) =&\mathbb{E}\left[Y_i - \sum_{l\in D}\hat{\beta}_{i,l}(\pi)X_i^l\right]^2\\ =&\sigma_\epsilon^2 + \left(\sum_{l\in D}\beta_{i,l}X_i^l - \mathbb{E}[\sum_{l\in D}\hat{\beta}_{i,l}(\pi)X_i^l]\right)^2 + \mathbb{E}\left[\sum_{l\in D}\hat{\beta}_{i,l}(\pi)X_i^l - \mathbb{E}[\sum_{l\in D}\hat{\beta}_{i,l}(\pi)X_i^l]\right]^2,\end{aligned} \tag{19}$$

where the first term is referred to as irreducible error, the second term as bias, and the third term as variance. It is a standard result in the statistics and econometrics literatures that OLS is an unbiased estimator, that is,

$$\sum_{l\in D}\beta_{i,l}X_i^l - \mathbb{E}[\sum_{l\in D}\hat{\beta}_{i,l}(\pi)X_i^l] = 0, \text{ for } \pi = \{Agg, Dec\}. \tag{20}$$

We next quantify the variance term, which we denote by $\mathsf{Var}(\pi)$ for the estimation algorithm π. For the training data of item 1, we use the $m\times d$ matrix $X_1(\mathsf{tr}) = (X_{1,j}^l(\mathsf{tr}) : 1\le l\le d, 1\le j\le m)$ as the feature matrix, and the m dimensional vector $Y_1(\mathsf{tr}) := (Y_{1,j}(\mathsf{tr}) : 1\le j\le m)$. Analogously, for the training data of item 2, we use the $(\tau m)\times d$ matrix $X_2(\mathsf{tr}) = (X_{2,j}^l(\mathsf{tr}) : 1\le l\le d, 1\le j\le \tau m)$ as the feature matrix, and the τm dimensional vector $Y_2(\mathsf{tr}) := (Y_{2,j}(\mathsf{tr}) : 1\le j\le \tau m)$. Without loss of generality, we assume that the first d_s features are at the aggregate level, and the rest are at the individual level. Accordingly, for item 1, we denote by $X_1^s(\mathsf{tr}) = (X_{1,j}^l(\mathsf{tr}) : 1\le l\le d_s, 1\le j\le m)$ the feature matrix at the aggregate level, and by $X_1^n(\mathsf{tr}) = (X_{1,j}^l(\mathsf{tr}) : d_s+1\le l\le d, 1\le j\le m)$ the feature matrix at the individual level. For item 2, we denote by $X_2^s(\mathsf{tr}) = (X_{1,j}^l(\mathsf{tr}) : 1\le l\le d_s, 1\le j\le \tau m)$ the feature matrix at the aggregate level, and by $X_2^n(\mathsf{tr}) = (X_{1,j}^l(\mathsf{tr}) : d_s+1\le l\le d, 1\le j\le \tau m)$ the feature matrix at the individual level. For $i = 1, 2$, the OLS estimate of item i is given by:

$$\begin{aligned}\hat{\beta}_i(Dec) =&(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T Y_i(\mathsf{tr})\\ =&(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T (X_i(\mathsf{tr})\beta_i + \epsilon_i)\\ =&\beta_i + (X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T\epsilon_i,\end{aligned} \tag{21}$$

where the first equality follows from $Y_i(\mathsf{tr}) = X_i(\mathsf{tr})\beta_i + \epsilon_i$. We denote the sample size of items 1 and 2 by $m_1 = m$ and $m_2 = \tau m$, respectively. We are now ready to evaluate the variance of the decentralized model:

$$\begin{aligned}\mathsf{Var}_i(Dec) =&\mathbb{E}[\sum_{l\in D}\hat{\beta}_{i,l}(Dec)X_i^l - \mathbb{E}[\sum_{l\in D}\hat{\beta}_{i,l}(Dec)X_i^l]]^2\\ =&\mathbb{E}[(X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T)\epsilon_i\epsilon_i^T(X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T)^T]\\ =&\mathbb{E}[\mathbb{E}[(X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T)\epsilon_i\epsilon_i^T(X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T)^T|X_i, X_i(\mathsf{tr})]]\\ =&\sigma_\epsilon^2\mathbb{E}[(X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T)(X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i(\mathsf{tr})^T)^T]\\ =&\sigma_\epsilon^2\mathbb{E}[X_i^T(X_i(\mathsf{tr})^T X_i(\mathsf{tr}))^{-1}X_i] = \frac{\sigma_\epsilon^2}{m_i}\mathbb{E}\Big[X_i^T\left(\frac{1}{m_i}X_i(\mathsf{tr})^T X_i(\mathsf{tr})\right)^{-1}X_i\Big],\end{aligned}$$

where the second equality follows from Equation (21), the third from the law of iterated expectations, and the fourth from the fact that $\epsilon_{i,j}$ are *i.i.d.* with mean 0 and variance σ_ϵ^2. By the strong law of large numbers, we have $\lim_{m_i\uparrow+\infty} \frac{1}{m_i} X_i(\mathsf{tr})^T X_i(\mathsf{tr}) = Id_d$, where Id_d is the identity matrix with dimension d, observing that the features are *i.i.d.* distributed with mean 0 and variance 1. Therefore, we have

$$\lim_{m_i\uparrow+\infty} m_i \cdot \mathsf{Var}_i(Dec) = \sigma_\epsilon^2 \mathbb{E}[X_i^T X_i] = \sigma_\epsilon^2 \cdot d (= \sigma_\epsilon^2 \cdot (d_s + d_n)),$$

where the second equality follows from the fact that X_i has d features which are *i.i.d.* with mean 0 and variance 1. Thus, for item 1, we obtain

$$\lim_{m\uparrow+\infty} m \cdot \mathsf{Var}_1(Dec) = \sigma_\epsilon^2 (d_s + d_n), \tag{22}$$

and, for item 2,

$$\lim_{m\uparrow+\infty} m \cdot \mathsf{Var}_2(Dec) = \frac{\sigma_\epsilon^2 (d_s + d_n)}{\tau}. \tag{23}$$

We next analyze the aggregate model. The true data-generating process (of the training set) can be specified as: $Y = X(\mathsf{tr})\beta + \epsilon$, where Y is a $(1+\tau)m$-dimensional outcome vector that concatenates Y_1 and Y_2, ϵ is a $(1+\tau)m$-dimensional error vector that concatenates ϵ_1 and ϵ_2, and $X(\mathsf{tr})$ is the $(1+\tau)m \times (d_s + 2d_n)$-dimensional feature matrix defined as follows:

$$X(\mathsf{tr}) := \begin{pmatrix} X_1^s(\mathsf{tr}), & X_1^n(\mathsf{tr}), & 0 \\ X_2^s(\mathsf{tr}), & 0, & X_2^n(\mathsf{tr}) \end{pmatrix}.$$

Thus, β is a $(d_s + 2d_n)$-dimensional vector where the first d_s entries are the coefficients of the features at the aggregate level, the next d_n entries are the coefficients of the features at the individual level for item 1, and the last d_n entries are the coefficients of the features at the individual level for item 2. As a result, for the aggregate model, the coefficient estimates are given by:

$$\hat{\beta}(Agg) = (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T Y(\mathsf{tr}) = \beta + (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T \epsilon. \tag{24}$$

To evaluate $\mathsf{Var}_i(Agg)$, we define two auxiliary $(d_s + 2d_n)$-dimensional random vectors $\tilde{X}_1 = (X_1^T, \mathbf{0})^T$ and $\tilde{X}_2 = ((X_2^s)^T, \mathbf{0}, (X_2^n)^T)^T$, where X_i $(i = 1, 2)$ are defined in the same way as in the decentralized model, $\mathbf{0}$ is a d_n-dimensional 0 vector, $X_2^s = (X_2^1, X_2^2, ..., X_2^{d_s})$ is the aggregate-level part of X_2, and $X_2^n = (X_2^{d_s+1}, X_2^{d_s+2}, ..., X_2^{d_s+})$ is the individual-level part of X_2. We are now ready to compute $\mathsf{Var}_i(Agg)$:

$$\begin{aligned}
&\mathsf{Var}_i(Agg) = \mathbb{E}[\sum_{l\in D} \hat{\beta}_{i,l}(Agg) X_i^l - \mathbb{E}[\sum_{l\in D} \hat{\beta}_{i,l}(Agg) X_i^l]]^2 \\
=&\mathbb{E}[(\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T) \epsilon \epsilon^T (\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T)^T] \\
=&\mathbb{E}[\mathbb{E}[(\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T) \epsilon \epsilon^T (\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T)^T | \tilde{X}_i, X(\mathsf{tr})]] \\
=&\sigma_\epsilon^2 \mathbb{E}[(\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T)(\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} X(\mathsf{tr})^T)^T] = \sigma_\epsilon^2 \mathbb{E}[\tilde{X}_i^T (X(\mathsf{tr})^T X(\mathsf{tr}))^{-1} \tilde{X}_i] \\
=&\sigma_\epsilon^2 \mathbb{E}\Big[((X_i^s)^T, (X_i^n)^T) \begin{pmatrix} X_1^s(\mathsf{tr})^T X_1^s(\mathsf{tr}) + X_2^s(\mathsf{tr})^T X_2^s(\mathsf{tr}), & X_i^s(\mathsf{tr})^T X_i^n(\mathsf{tr}) \\ X_i^n(\mathsf{tr})^T X_i^s(\mathsf{tr}), & X_i^n(\mathsf{tr})^T X_i^n(\mathsf{tr}) \end{pmatrix}^{-1} \begin{pmatrix} X_i^s \\ X_i^n \end{pmatrix}\Big] \\
=&\frac{\sigma_\epsilon^2}{(1+\tau)m} \mathbb{E}\Big[((X_i^s)^T, (X_i^n)^T) \begin{pmatrix} \frac{1}{(1+\tau)m}(X_1^s(\mathsf{tr})^T X_1^s(\mathsf{tr}) + X_2^s(\mathsf{tr})^T X_2^s(\mathsf{tr})), & \frac{1}{(1+\tau)m} X_i^s(\mathsf{tr})^T X_i^n(\mathsf{tr}) \\ \frac{1}{(1+\tau)m} X_i^n(\mathsf{tr})^T X_i^s(\mathsf{tr}), & \frac{1}{(1+\tau)m} X_i^n(\mathsf{tr})^T X_i^n(\mathsf{tr}) \end{pmatrix}^{-1} \begin{pmatrix} X_i^s \\ X_i^n \end{pmatrix}\Big],
\end{aligned}$$

where the second equality follows from Equation (24), the third from the law of iterated expectations, the fourth from the fact that $\epsilon_{i,j}$ are *i.i.d.* with mean 0 and variance σ_ϵ^2, and the sixth from the definitions of $\tilde{X}_i$ and $X(\mathsf{tr})$. By the strong law of large numbers, we have

$$\lim_{m\uparrow+\infty} \frac{1}{(1+\tau)m}[X_1^s(\mathsf{tr})^T X_1^s(\mathsf{tr}) + X_2^s(\mathsf{tr})^T X_2^s(\mathsf{tr})] = Id_{d_s},$$

where Id_{d_s} is the identity matrix with dimension d_s, observing that the features are *i.i.d.* with mean 0 and variance 1. Analogously, observing that

$$\gamma_i := \frac{m_i}{(1+\tau)m} = \begin{cases} \frac{1}{1+\tau}, & \text{for } i=1, \\ \frac{\tau}{1+\tau}, & \text{for } i=2, \end{cases}$$

we obtain

$$\lim_{m\uparrow+\infty} \frac{1}{(1+\tau)m} X_i^s(\mathsf{tr})^T X_i^n(\mathsf{tr}) = \frac{m_i}{(1+\tau)m} \cdot \lim_{m\uparrow+\infty} \frac{1}{m_i} X_i^s(\mathsf{tr})^T X_i^n(\mathsf{tr}) = \mathbf{0}_{d_s \times d_n},$$

and

$$\lim_{m\uparrow+\infty} \frac{1}{(1+\tau)m} X_i^n(\mathsf{tr})^T X_i^n(\mathsf{tr}) = \frac{m_i}{(1+\tau)m} \cdot \lim_{m\uparrow+\infty} \frac{1}{m_i} X_i^n(\mathsf{tr})^T X_i^n(\mathsf{tr}) = \gamma_i \cdot Id_{d_n} = \begin{cases} \frac{1}{1+\tau} \cdot I_{d_n}, & \text{for } i=1, \\ \frac{\tau}{1+\tau} \cdot I_{d_n}, & \text{for } i=2. \end{cases}$$

Therefore, we have

$$\lim_{m\uparrow+\infty} m \cdot \mathsf{Var}_i(Agg) = \frac{\sigma_\epsilon^2}{1+\tau} \cdot \mathbb{E}[(X_i^s)^T X_i^s] + \frac{\sigma_\epsilon^2}{(1+\tau)\gamma_i} \cdot \mathbb{E}[(X_i^n)^T X_i^n] = \sigma_\epsilon^2 \cdot \left(\frac{d_s}{1+\tau} + \frac{d_n}{(1+\tau)\gamma_i} \right),$$

where the second equality follows from the fact that X_i has d features which are *i.i.d.* with mean 0 and variance 1. Thus, for item 1, we obtain

$$\lim_{m\uparrow+\infty} m \cdot \mathsf{Var}_1(Agg) = \sigma_\epsilon^2 \cdot \left(\frac{d_s}{1+\tau} + d_n \right), \tag{25}$$

and, for item 2,

$$\lim_{m\uparrow+\infty} m \cdot \mathsf{Var}_2(Agg) = \sigma_\epsilon^2 \cdot \left(\frac{d_s}{1+\tau} + \frac{d_n}{\tau} \right). \tag{26}$$

Plugging (20), (22), (23), (25), and (26) into (19), we have, for item 1,

$$\lim_{m\uparrow+\infty} m \cdot [\mathsf{e}_1(Dec) - \mathsf{e}_1(Agg)] = \lim_{m\uparrow+\infty} m \cdot [\mathsf{Var}_1(Dec) - \mathsf{Var}_1(Agg)] = \left(1 - \frac{1}{1+\tau}\right) \cdot d_s \cdot \sigma_\epsilon^2,$$

and, for item 2,

$$\lim_{m\uparrow+\infty} m \cdot [\mathsf{e}_2(Dec) - \mathsf{e}_2(Agg)] = \lim_{m\uparrow+\infty} m \cdot [\mathsf{Var}_2(Dec) - \mathsf{Var}_2(Agg)] = \left(\frac{1}{\tau} - \frac{1}{1+\tau}\right) \cdot d_s \cdot \sigma_\epsilon^2.$$

This concludes the proof of Proposition 4. □